Technical Report

IISc/SID/AE/AOARD/2013/01

Reactive Collision Avoidance of UAVs with Stereovision Sensing

Amit Kumar Tripathi
Ph.D. Student

Ramsingh G. Raja
Project Associate

Radhakant Padhi
Associate Professor

Department of Aerospace Engineering
Indian Institute of Science
Bangalore, India.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **17 JAN 2014**	2. REPORT TYPE **Final**	3. DATES COVERED **09-08-2011 to 08-08-2013**

4. TITLE AND SUBTITLE **Collision Avoidance guidance of UAVs with Obstacle Estimation from Vision Sensing**	5a. CONTRACT NUMBER **FA23861114096**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) **Radhakant Padhi**	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Indian Institute of Science,Indian Institute of Science,Bangalore 560-012,India,IN,560-012**	8. PERFORMING ORGANIZATION REPORT NUMBER **N/A**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) **AOARD, UNIT 45002, APO, AP, 96338-5002**	10. SPONSOR/MONITOR'S ACRONYM(S) **AOARD**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) **AOARD-114096**

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Using simple passive stereovision pin-hole sensors, an effective reactive collision avoidance algorithm is presented in this report for unmanned aerial vehicle (UAV). Both kinematic model as well as point mass models of UAV are considered for system dynamics. The stereovision camera is considered mounted on the UAV. The camera sensors and the obstacle are forming a geometrical configuration and triangulation methods are used to estimate the position and velocity of the obstacle falling under the field of view of the camera sensors. For simplicity, shape of the obstacle is considered as spherical. The position estimation of the obstacle is also carried out by image processing methods wherein the feature points detection, stereo matching and triangulation algorithm are used to compute the 3D reconstruction of obstacle. This method uses the computer vision system toolbox and VRML viewer of MATLAB r, Using this we are able to compute the approximate size and centre of the obstacle. This geometrical formulation takes advantage of both 'stereo vision' as well as 'optical flow' signatures and hence it is capable of estimating the range information as well, making its position estimate quite accurate. A large number of simulation studies, which also includes consistency checks for filtering algorithms, leads to the conclusion that this strategy is quite effective in avoiding popup obstacles within a very short time and hence can be very useful for reactive collision avoidance.

15. SUBJECT TERMS

Aircraft Control, Flight Control, UAV

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**104**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Table of Contents

List of Figures

Abstract

Using simple passive stereovision pin-hole sensors, an effective reactive collision avoidance algorithm is presented in this report for unmanned aerial vehicle (UAV). Both kinematic model as well as point mass models of UAV are considered for system dynamics. The stereovision camera is considered mounted on the UAV. The camera sensors and the obstacle are forming a geometrical configuration and triangulation methods are used to estimate the position and velocity of the obstacle falling under the field of view of the camera sensors. For simplicity, shape of the obstacle is considered as spherical. The position estimation of the obstacle is also carried out by image processing methods wherein the feature points detection, stereo matching and triangulation algorithm are used to compute the 3D reconstruction of obstacle. This method uses the computer vision system toolbox and VRML viewer of MATLAB ®, Using this we are able to compute the approximate size and centre of the obstacle. The camera sensors are considered noisy and the noise is approximated by a gaussian random variable. First, an Extended Kalman Filtering (EKF) approach is proposed to extract the useful information from the noisy information generated. Further Unscented Kalman Filter (UKF) approach is proposed for capturing the camera sensor nonlinearity in a better manner as UKF enables a second order approximation. This geometrical formulation takes advantage of both 'stereo vision' as well as 'optical flow' signatures and hence it is capable of estimating the range information as well, making its position estimate quite accurate. Further the obstacle velocity information is also estimated based on optical flow information. Next, an 'aiming point' is computed after putting an artificial safety ball around the obstacle and using the collision cone approach. Next, the velocity vector of the vehicle is steered away towards this aiming point using a recently developed 'differential geometry guidance' which is a dynamic inversion based three-dimensional nonlinear aiming point guidance law. The guidance command generation is based on angular correction between the actual and the desired direction of the velocity vector. Note that the velocity vector gets aligned along the selected aiming point quickly (i.e. within a fraction of the available time-to-go), which makes it possible to avoid pop-up obstacles. A large number of simulation studies, which also includes consistency checks for filtering algorithms, leads to the conclusion that this strategy is quite effective in avoiding popup obstacles within a very short time and hence can be very useful for reactive collision avoidance. The guidance algorithm has been verified with simulations carried out both for single obstacles as well as for multiple obstacles on the path, stationary as well as moving obstacles and also with different safety sphere sizes around the obstacles. The proposed algorithm has been validated using both kinematic as well as point mass model of a prototype unmanned aerial vehicle. For better confidence, results have also been validated by incorporating a first order autopilot delay compensation for control commands.

Chapter 1

Introduction

UAV's are used for many missions of practical importance due to their ability to fly autonomously at low altitudes. The UAVs can be deployed in numerous fields. The major areas of applications are reconnaissance and surveillance, targeted attacks with less collateral damage, battle damage assessment, traffic monitoring for crime prevention, detection and containment of hazardous leakages in industries, assessment and rehabilitation in case of natural calamities, military and terrain mapping etc. While operating at low altitude, the UAV may come across many obstacles on it's path, some of them are well known such as urban structures, buildings, trees etc. Other obstacles may suddenly appear as pop-up threat across it's path. The UAV's are generally equipped with path planning algorithms that take care of all known obstacles and accordingly the trajectory up to destination is well planned. The pop-up threats are unpredictable, therefore all possible collision avoidances cannot be accounted a-priori, hence, it is vital that UAV's should have the capability to cater for all kinds of collision avoidance with obstacles autonomously and safely reach to it's destination. A variety of path planning algorithms are available which are used to design the trajectory of a UAV. Generally these algorithms are computed off-line and are stored in UAV onboard processor. However, there is no such off-line mechanism for guiding the UAV in a scenario when the obstacles suddenly appear as a pop-up threat across the UAV trajectory. Therefore, there is an immense need of an online algorithm that is built-in the processor to compute the alternative path for the UAV satisfying it's own constraints and safely protect it from colliding in a given minimum available time. The online algorithm that is computationally fast, highly efficient and noninteractive in nature serves such kind of purpose and are called as reactive collision avoidance algorithm. The trajectory described in the Fig.1.1 shows the reactive collision avoidance for the pop-up threat.

The collision avoidance problem addressed in this report can in fact be broadly termed into the problem of 'path planning', which is the process of finding a safe flight path to the goal point. It typically consists of two layers (i) a global path planner and (ii) a local path planner. A global path planning algorithm accounts for all known obstacles in the domain a-priori and plans a path to the destination in such a way that it avoids all obstacles on its path. Some global path planning algorithms such as rapidly exploring random tree (RRT), Potential field method and graph search methods have been proposed in the literature for this task. An interested reader can see [6] for an overview of these methods. This path planning is deterministic in nature. For local path planning, however, the main aim is to avoid collisions with obstacles at close vicinity (especially with the pop-up threats), which is done in the following manner. When an onboard sensor detects obstacle(s) and predicts a possible collision, the guidance law attempts to maneuver the vehicle away from the danger as soon as possible so that the impending collision is averted. Such a guidance is also called as 'reactive guidance', since the available reaction time is very small and decisions have to be taken quickly (i.e. within a fraction of the available time-to-go). Note that reactive collision avoidance guidance typically results in high manoeuvres for a small duration of time. Moreover, the guidance law should also ensure that while avoiding obstacles the vehicle should not deviate too much from its original intended path. This is both because other obstacles should not come on its new flight path and it should not deviate too much away from seeking its intended destination (otherwise, putting it back into track becomes difficult). Another critical restriction on such a reactive guidance law is that it should be computationally quite efficient (preferably available in closed form), since the time to react is typically quite low and, if possible, the control and guidance update needs to be done at a higher rate. Moreover, for fixed-wing UAVs, it should also assure that the vehicle keeps moving at sufficiently higher velocity to avoid stalling. In view of these, there is a need for developing a reactive guidance scheme based on sound geometric and mathematical considerations. Preferably it should also have sufficient generality so that it can be applied for a wide range of applications.

It is vital that UAV should fly autonomously to sense and avoid collisions [38]. Environment sensing is generally achieved through aboard passive vision sensing through onboard

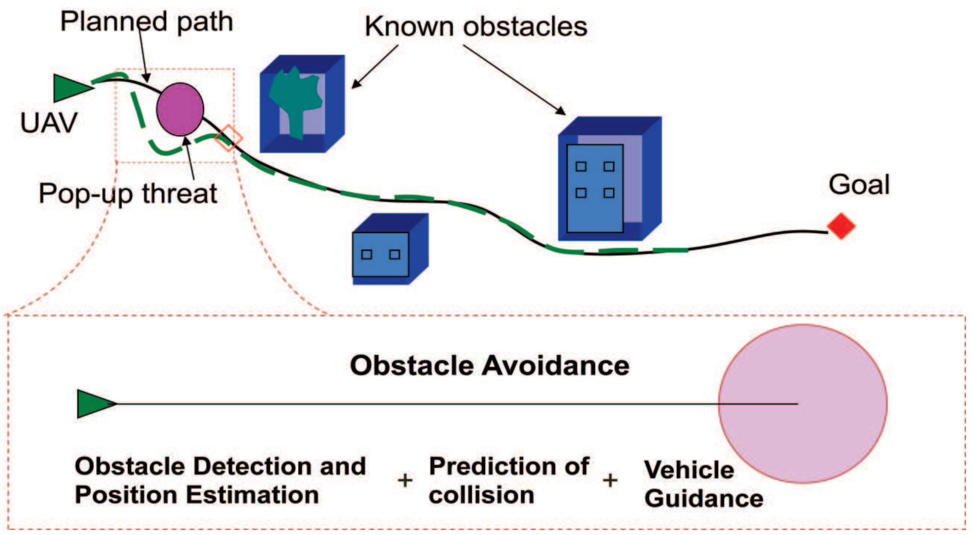


Figure 1.1: Obstacle Collision Avoidance

cameras. It has been widely implemented due to their light weight, low cost and energy usage[11]. When obstacles are sensed in the environment, UAV must be able to react and maneuver quickly so that the collision is averted. Because of the fact that the time availability is small, the collision avoidance guidance algorithm should also be computationally efficient (preferably should be computed in closed form). To achieve this, an algorithm needs to be designed which steers the vehicle to avert the obstacle as well as plan the vehicle's path as fast enough to be implemented online. Such algorithms are called "reactive obstacle avoidance algorithm". These algorithm may sustain the challenge of heavy computational limit imposed by UAV flying at high speeds. Apart from speed, an important requirement is low computational resource usage, so that it may be suitable for onboard execution. Many global path planning algorithms are computationally expensive, hence they are inadequate to be solved in the limited memory usage [12]. Hence, there is a urgent need of new technique which is preferably based on sound geometric and mathematical considerations so that, due its generality and scalability, it can be applied for wide range of applications.

In this report, a reactive collision-avoidance algorithm is proposed, the UAV invokes the reactive collision-avoidance algorithm only when an obstacle is encountered by onboard stereovision sensors across its trajectory. The collision avoidance occurs in two steps; first, the collision is detected based on the collision cone approach [41, 17] using vision sensor data. Next, if the threat is observed, a collision avoidance maneuver is performed with respect to an 'aiming point'[34], judiciously selected on the obstacle's safety sphere surface. Note that the concept can be interpreted in the light of 'pursuit guidance', where the objective is to reorient or align the velocity vector of the UAV to the line-of-sight [33]. In all the cases, the gains are selected such that the obstacles are avoided quickly (i.e. within a fraction of the available time-to-go).

Vision sensors are finding its usefulness as an aid for navigation and guidance systems for UAVs in many potential operations. Since such sensors are usually cheap, light-weight, fairly reliable and more importantly they are passive in nature. A passive sensor is not susceptible to signal jamming either. It can also enable the vehicle in GPS denied environments as well as in the areas where the navigation information is not available on the trajectory of UAV. However, there are several drawbacks as well, which are the noisy nature of the signal, the lack of range information and poor resolution. However, a single passive vision sensor is only capable to provide direction information of the obstacle in general. Unless the relative velocity between the vehicle and obstacle is high, the range signature is very weak (it is zero if there is no relative velocity) and hence it can not be estimated well. To address this drawback, a stereo vision sensor based formulation is presented in this report, where the geometric constraint serves as a 'virtual sensing' of the range.

Vision sensors capture the projection of the obstacle environment on their image plane and the data frames obtained by the vision sensor is processed by onboard image processor to find the information about the obstacle. The practical stereo camera models available are noisy in nature. Therefore information gathered by the Vision sensor is not accurate. Which if used directly can give erroneous results. In order to deal with such issues filtering algorithms such as an Extended Kalman Filter (EKF) [5, 3] and Unscented Kalman Filter [3, 4] are used which tries to extract the useful information from the noisy data.

A local path planning algorithm attempts to avoid unforeseen obstacles which suddenly appear on its flight path in close proximity. The time-to-go in such problems is usually small and the vehicle has to take quick corrective action of its flight path within the available short time. It is often called as 'reactive collision avoidance'. Since the onboard processor installed in UAVs have a very limited computing capability, such an algorithm needs to be computationally very efficient. Using an innovative 'collision cone' approach, such an algorithm was first proposed in [41] and the philosophy has subsequently been augmented with 3D prospective [17]. Using this idea and fusing it with the 'aiming point guidance' [8], a ‘differential geometry guidance’ algorithm has been proposed recently in [7], which has been used in this report as well. Note that such an approach has also been proposed earlier in [9], but it was only with a single camera formulation and hence was only moderately successful. The approach presented here, however, involved a two camera formulation and hence turns out to be much more effective because of better estimation of the obstacle position [10] and velocity. A large number of simulation studies, which include σ-bound consistency checks of EKF as well as UKF, clearly brings out this fact.

Simulation studies have been carried out with the ’Kinematic model’ as well as the ’Point mass model’ of the real fixed wing UAV to test the performance of the nonlinear differential geometric guidance scheme. Various cases have been considered with one and multi-obstacle scenario, different safety radius of the obstacle, stationary as well as moving obstacles in noise-free and noisy conditions. For all the cases, the UAV avoids the obstacle and reaches the goal point. Results are also validated by incorporating first order autopilots for the guidance commands in case of point mass dynamics of UAV.In the present work, the problem of reactive obstacle avoidance for UAV is addressed by collision cone approach using differential geometric guidance and forming a geometrical configuration with a stereo camera model.

1.1 Motivation

The problem of reactive collision avoidance for UAV has been heavily researched in recent literature. The artificial potential field method [13] is a popular approach due to its intuitive nature and capability to be tailored to different types of problems. The potential fields in obstacle avoidance are tailored such that obstacles have a repulsive field while the destination has an attractive field. The resultant field represents a safe direction for the UAV to move along. However, this algorithm is not strictly reactive, since at every instant it takes into account the presence of all (or most of) the obstacles in the environment before deciding the direction. A model-predictive control (MPC) based collision avoidance algorithm is proposed [14], in which a potential field function is incorporated in the cost function to be minimized. The other terms in the cost function include costs for path following, control saturation and input saturation. The advantage of using MPC is that state and input constraints are accounted. The disadvantage of such a strategy is that the algorithm functions under rigid path following requirements and does not actively seek the destination. Further, MPC is a resource-intensive algorithm that requires a powerful processor, making the method unsuitable for implementation aboard a UAV. Another approach in reactive collision avoidance is RRT [12], which is a randomized search algorithm. In RRT algorithm the length of the path found is far from optimal and may have several extraneous branches due to the random nature of the algorithm. Although reactive collision avoidance permits maneuvers that are not optimal but the wastage in the path found by RRT is significant. However, a path pruning algorithm can refine the path but such a step is infeasible for online collision avoidance. Some graph search algorithms like the best-first search algorithm are implemented for reactive collision avoidance [15]. In the best-first search method a sorted list of pre-computed motion primitives are created. Saving the pre-computing motion primitives in a lookup table is infeasible for UAV applications due to the large memory resources demanded.

Even the problem of UAV pursuing its goal is also a similar approach, which has been implemented with intermediate obstacles in the path using PN guidance based collision avoidance scheme [16]. However, this scheme leads to a jump in the control effort every time a new target is pursued. Instead, a minimum effort guidance (MEG) approach minimizes the control effort for the entire trajectory along with avoiding collisions for multiple targets [17]. A collision cone approach [41] is used to detect potential collisions by considering a threat boundary around the obstacle in MEG guidance. It has been demonstrated that MEG is more suitable than PN [17]. However, collision avoidance problems do not have minimum effort requirements and emphasize vehicle safety over low control effort. The MEG guidance causes the vehicle to maneuver until the point of impact, which is risky. Above all, the collision avoidance algorithm must ensure that the UAV full dynamics should be accounted. In some of the literature, obstacle avoidance issues are addressed through the kinematic model, [18]-[19] in which the autopilot responses are approximated by first order models. Even $3-DOF$ motion is also considered to some extent [17], [20]. This may cause vehicle to take large and practically infeasible maneuvers, leading to state or control saturation.

1.2 Contribution

In the present work, a new method of computing the aiming point is developed. The philosophy is based on collision cone approach. The aiming point is the point where the vehicle has to quickly maneuver in order to avoid an imminent collision. The philosophy behind this approach is that the velocity vector is projected to a plane which intersects the obstacle safety sphere if the point of projection lies inside the obstacle safety sphere then the aiming point guidance is invoked and a new aiming point is computed which demands deflection in UAV velocity so that collision can be averted. The differential geometric guidance is used to compute the guidance command for driving the UAV to aiming point as well as to the destination.

In the present work, a reactive obstacle avoidance algorithm is designed which has been realized with kinematic model as well as point mass model of a realistic UAV. The nonlinear differential geometric guidance (DGG) algorithm is First, successfully implemented with kinematic model and Next, validated with the point mass model based formulation and simulations are carried out with randomized input conditions. The guidance algorithm detects the obstacle based on the $3D$ collision cone approach [17] and the avoidance maneuver is performed. The nonlinear guidance algorithm generates angular commands in the horizontal and the vertical plane. These commands are then pursued by the UAV to reach the aiming point [33],[34]. The aiming point is the point of contact of the tangent drawn through the UAV location to the safety ball skirting the obstacle. The tangent is the line of sight of the vehicle to the aiming point. The concept is implemented in the direction of the pursuit guidance [33] where the objective is to reorient the velocity vector of the vehicle to the line of sight (LOS) within the fraction of the available time-to-go. It finally steers UAV towards the aiming point and hence averts the obstacle. Pursuit guidance/aiming point guidance [33],[34] philosophy is used in the missile guidance to aim at the predicted position of the target at the final time.

The algorithm is successfully realized with noise-free sensor as well as plant model of UAV. In order to test the performance in noisy conditions, simulations are carried out with noisy sensor as well as plant model. The estimation techniques are used to extract the useful information from the noisy data. The reactive collision avoidance algorithm is successfully realized under noisy conditions.

In the present work, a recently developed Nonlinear differential geometric guidance (DGG) is used for reactive collision avoidance. The obstacles considered in the problem are moving as well as stationary. The geometric configuration is formed with stereovision pin hole camera sensors mounted on the UAV. The camera sensors capture the projection of the obstacle on image plane. Using the camera coordinate information of the pixel as well as the sequential pixel frames, one can compute the obstacle position as well as obstacle velocity within prescribed tolerance bounds.

Simulation studies have been carried out with different plant models to test the performance of the innovative nonlinear reactive guidance scheme elaborately. Scenarios with different number and size of the obstacles in the environment have been considered. Point mass model with a coordinated flight is demonstrated with all scenarios, where the controller dynamics is approximated by the first order autopilots[20].

Various simulation studies clearly show that, using the stereo camera sensors in the specific geometrical formulation and using the new methodology of the aiming point computation, the reactive collision avoidance for stationary as well as moving obstacles is performed for the UAV model. The aiming point is achieved by applying the nonlinear differential geometric guidance law, which maneuvers the UAV quickly in available time-to-go. The technique is quite effective in avoiding collisions in different scenarios. In all the simulations, all the constraints posed by the vehicle capability are very well met within the available time-to-go.

Chapter 2

Obstacle Position and Velocity Estimation

In this section, stereo geometry formulation with two pin-hole cameras and an obstacle, falling under field of view (fov) of both cameras, has been discussed. Here, the objective is to compute the position of the obstacle using the geometry formulation. The obstacle is considered as a point. The projection of this point obstacle on the camera image planes are processed through the onboard image processors and the corresponding camera coordinates are assumed to be known. The position of the obstacle can be obtained using triangulation method using the data obtained from camera sensors. However, the vision sensors are assumed to be noisy. So, we need to estimate exact position of the obstacle using filtering technique such as Extended kalman filter(EKF) and Unscented kalman filter(UKF) which are discussed in later part of this section. Due to the sensor noise, there is uncertainty in exact position estimation. Therefore, the safety of the point obstacle is insured by assuming the obstacle at the center of an sphere and the sphere is treated as virtual obstacle and calling the radius of sphere as safety radius of the point obstacle. Therefore, the sphere with finite safety radius is eventually termed as obstacle with safety bounds. The center and radius of this spherical obstacle can be computed by image processing techniques by capturing the spherical obstacle image through camera vision sensors.

2.1 Stereo Geometry

The geometrical formulation [10] for computing the position of the obstacle with stereo vision sensors is considered as per Fig. 2.1.

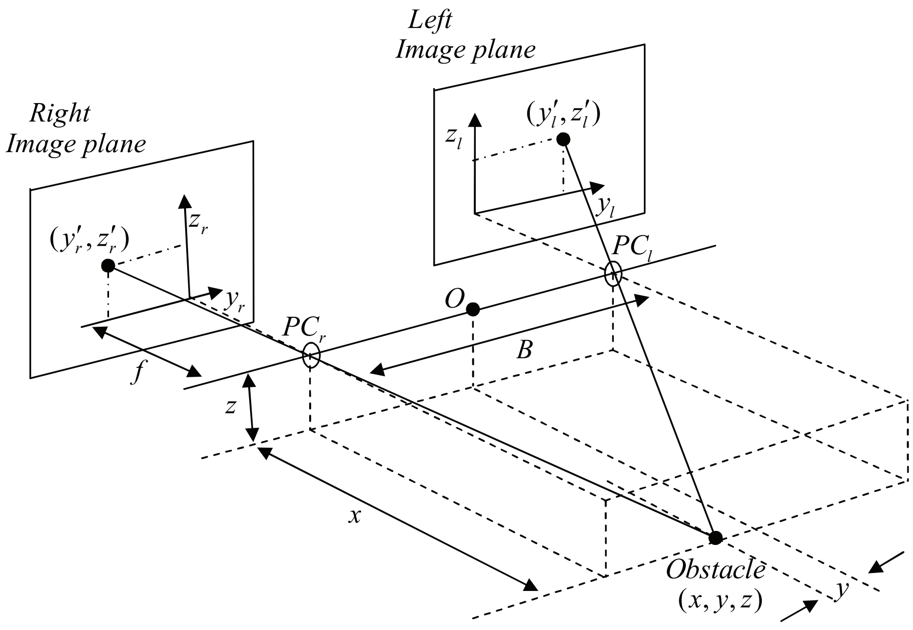


Figure 2.1: Geometry Formulation with Two Pin-hole Camera

The pin-hole cameras (PC_r and PC_l) are fitted on the UAV are symmetric with respect to longitudinal axis as well as center of the gravity O. Both camera's optical axis are parallel and separated by baseline distance B. The focal length for both the cameras is same and is denoted by f. Cameras are assumed to be non-rotating. The y axis of the $3D$ coordinate system is parallel to the baseline. The obstacle is located at point (x, y, z) should be visible to both the cameras for range information extraction. The range computation of the obstacle is determined based on the baseline separation, which should be more for faraway obstacle. The image coordinates corresponding to the obstacle for left camera is $(y_l^{'}, z_l^{'})$ and for the right camera is $(y_r^{'}, z_r^{'})$. The range information of the obstacle is computed from the stereo images by obtaining (θ_l, ϕ_l) from left camera and (θ_r , ϕ_r) from right camera. Where θ_l, θ_r are the azimuth angle and ϕ_l, ϕ_r are the elevation angle of obstacle on the image plane of both cameras.

Using the triangle rule for camera coordinate frame

$$y_l' = f\ tan(\theta_l) \tag{2.1}$$

$$y_r' = f\ tan(\theta_r) \tag{2.2}$$

$$z_l' = \sqrt{[f^2 + (y_l')^2]}\ tan(\phi_l) \tag{2.3}$$

$$z_r' = \sqrt{[f^2 + (y_r')^2]}\ tan(\phi_r) \tag{2.4}$$

Using similar triangles,

$$\frac{y_l'}{-f} = \frac{(-y + \frac{B}{2})}{x} \tag{2.5}$$

$$\frac{y_r'}{-f} = \frac{(-y - \frac{B}{2})}{x} \tag{2.6}$$

$$\frac{z_l'}{-f} = \frac{z_r'}{-f} = \frac{-z}{x} \tag{2.7}$$

The obstacle position can be computed with simple algebra on Eq.(2.5) to Eq. (2.7) as

$$x = -\frac{Bf}{(y_l' - y_r')} \tag{2.8}$$

$$y = -\frac{B(y_l' + y_r')}{2(y_l' - y_r')} \tag{2.9}$$

$$z = -\frac{B(z_l' + z_r')}{2(y_l' - y_r')} \tag{2.10}$$

Thus the obstacle position is a function of disparity which is defined as $(y_l' - y_r')$.

2.2 Obstacle Position Estimation using EKF

Position information of the obstacle is computed with sensor data obtained from vision sensing cameras. The vision sensors are assumed to be noisy. Moreover, the UAV model is considered with process noise as well. To address these issues, EKF [3] is used to filter out the noise and estimate the obstacle position accurately. An assumption made here is that the measurement noise uncertainty is within 10% of the actual value.

2.2.1 Dynamics of System

The camera sensor's nonlinear state dynamics as per state space model is given as follows

$$\dot{X}_r = f(X_r, t) + G(t)w(t) \tag{2.11}$$

$$Y_k = h(X_r(k)) + \nu_k \tag{2.12}$$

where camera sensor state propagation equation is in continuous time domain and output state measurement equation is in discrete time domain, where $f(X_r, t)$ is nonlinear transition matrix function and $h(X_r(k))$ is nonlinear measurement matrix function. $G(t)$ is process noise influence matrix, it is considered as $G(t) = I$. Process noise $w(t)$ and measurement noise ν_k are independent, zero mean, gaussian noise processes with their covariance matrix Q and R respectively. The mathematical expression is as follows

$$E[w(t)w^T(\tau)] = Q(t)\delta(t-\tau) \tag{2.13}$$

$$E[\nu_k \nu_j^T] = R\delta_{kj} \tag{2.14}$$

where $\delta(t-\tau)$ is dirac-delta function and δ_{kj} is kronecker delta function.

In EKF framework, linearize the nonlinear state space model at each instant around the current state estimate, therefore $f(X_r, t)$ and $h(X_r(k))$ are linearized around $\hat{X}_r$. Use the linearized model for state error covariance propagation. The state vector derivative $\dot{X}_r$ is defined as

$$\dot{X}_r = \begin{bmatrix} \dot{r}_l & \dot{\theta}_l & \dot{\phi}_l & \dot{r}_r & \dot{\theta}_r & \dot{\phi}_r \end{bmatrix}^T$$

where r_l, θ_l and ϕ_l are left camera states and r_r, θ_r and ϕ_r are right camera states. The dynamics of camera state vector is given as

$$\begin{bmatrix} \dot{r}_l \\ \dot{\theta}_l \\ \dot{\phi}_l \\ \dot{r}_r \\ \dot{\theta}_r \\ \dot{\phi}_r \end{bmatrix} = \begin{bmatrix} \cos\theta_l \cos\phi_l u_r + \sin\theta_l \cos\phi_l v_r + \sin\phi_l w_r \\ -\frac{\sin\theta_l}{r_l \cos\phi_l} u_r + \frac{\cos\theta_l}{r_l \cos\phi_l} v_r \\ -\frac{\cos\theta_l \sin\phi_l}{r_l} u_r - \frac{\sin\theta_l \sin\phi_l}{r_l} v_r + \frac{\cos\phi_l}{r_l} w_r \\ \cos\theta_r \cos\phi_r u_r + \sin\theta_r \cos\phi_r v_r + \sin\phi_r w_r \\ -\frac{\sin\theta_r}{r_r \cos\phi_r} u_r + \frac{\cos\theta_r}{r_r \cos\phi_r} v_r \\ -\frac{\cos\theta_r \sin\phi_r}{r_r} u_r - \frac{\sin\theta_r \sin\phi_r}{r_r} v_r + \frac{\cos\phi_r}{r_r} w_r \end{bmatrix} \tag{2.15}$$

where, u_r, v_r and w_r are relative velocity components of UAV along x, y and z directions.

2.2.2 Camera Sensor Measurement Noise Model

Camera Sensor measurement noise model is considered based on the fact that noise is proportional to sensing range as the sensing range increases uncertainty also increases. Therefore, measurement noise model is considered as per Eq.(2.16) where m_k is percentage noise at time instant k, m_0 is initial percentage measurement noise taken as 20, $r(k)$ is the range of obstacle at time instant k and δ is a tuning parameter considered as 0.99 which defines how m_k changes with $r(k)$

$$m_k = m_0 \left(1 - \delta^{r(k)}\right) \tag{2.16}$$

Measurement noise covariance can be computed based on m_k given as

$$M_k = \left(m_k \frac{w_v}{100} \times \frac{1}{3}\right)^2 \tag{2.17}$$

where w_v is the angular width of the camera field of view. It is assumed 120° for both horizontal and vertical axis.

2.2.3 Initialization

The camera state vector is initialized with first measurement of vision sensors

$$\hat{X}_r(0) = \begin{bmatrix} \hat{r}_l(0) & \hat{\theta}_l(0) & \hat{\phi}_l(0) & \hat{r}_r(0) & \hat{\theta}_r(0) & \hat{\phi}_r(0) \end{bmatrix}^T \tag{2.18}$$

The process noise covariance matrix Q is initialize as

$$Q = diag(\ 0.25,\ \ 0.028,\ \ 0.028,\ \ 0.25,\ \ 0.028,\ \ 0.028\ ) \tag{2.19}$$

The diagonal elements of Q corresponds to range $r(m)$, angles $\theta(radian)$ and $\phi(radian)$ of left and right cameras respectively. The error covariance matrix is initialized as

$$P_0 = diag\left(\begin{array}{cc} P_0^l, & P_0^r \end{array}\right) \tag{2.20}$$

where,

$$P_0^l = diag\left(\begin{array}{ccc} a_1 el^2, & a_2(\frac{m_0 w_v}{100})^2, & a_3(\frac{m_0 w_v}{100})^2 \end{array}\right) \tag{2.21}$$

$$P_0^r = diag\left(\begin{array}{ccc} a_1 er^2, & a_2(\frac{m_0 w_v}{100})^2, & a_3(\frac{m_0 w_v}{100})^2 \end{array}\right) \tag{2.22}$$

where P_0^l and P_0^r are the error covariance matrix and el and er are the corresponding initial error in the range of left and right camera respectively, $a_1 = 1$, $a_2 = 2$ and $a_3 = 2$ are tuning parameters.

2.2.4 Extended Kalman Filter

In EKF design, vision sensor model is nonlinear and the kalman filter is extended through linearization of nonlinear dynamics. The camera sensor states are estimated using linearization and applying extended kalman filter. In order to reduce errors due to linearization, the sensor true state should be close to estimated state at all time. Therefore the error dynamics can be represented by the linearized system about sensor's estimated state.

EKF is nonlinear recursive estimator. It linearizes the nonlinear dynamics around nominal operating point and operates around the narrow zone across the nominal operating point. The solution is recursive in the sense that each updated estimate of state is computed from previous state estimate and new measurement input data. Therefore only previous estimate of the state is needed and entire past state estimates need not be stored. Kalman filter is computationally more efficient than directly computing the state estimate from entire past observed data at each step of filtering process.

After linearization we first initialize the state estimate as well as state covariance matrix and then propagate the state estimation vector as $\hat{X}_r^+(k-1) \rightarrow \hat{X}_r^-(k)$: The following equation shows the state propagation which is as per state dynamics.

$$\dot{\hat{X}}_r = f(\hat{X}_r, t) \tag{2.23}$$

and propagate the covariance matrix $P_{k-1}^+ \rightarrow P_k^-$:

The following equation shows the state covariance propagation using the A matrix which is also known as system matrix of the linearized sensor state equation.

$$\dot{P}(t) = A(t)P(t) + P(t)A^T(t) + Q \tag{2.24}$$

The A matrix is computed by Linearizing Eq.(2.15). The mathematical expression for A is as follows

$$A(t) = \frac{\partial f}{\partial X_r}|_{\hat{X}_r(t)} \tag{2.25}$$

From Eq.(2.17), compute measurement error covariance matrix R_k is computed based on measurement noise model considered

$$R_k = diag(\ (M_k)_{\theta_l} \quad (M_k)_{\phi_l} \quad (M_k)_{\theta_r} \quad (M_k)_{\phi_r}\ ) \tag{2.26}$$

Compute Kalman gain K_k

$$K_k = P_k^- C_k^T [C_k P_k^- C_k^T + R_k]^{-1} \tag{2.27}$$

Linearizing $h(X_r(k))$ the matrix C_k is obtained , which is the measurement transition matrix. The C_k matrix is constant therefore it is not required to compute it for each iteration and saves computation time. It is used for computation of sensor output as described in Eq.(2.29).

$$C_k = \frac{\partial h}{\partial X_r}|_{\hat{X}_r^-(k)} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.28}$$

Take the measurement Y_k as

$$Y_k = C_k X_r + \nu_k \tag{2.29}$$

As the measurement from the camera sensor Y_k is obtained, update the camera sensor state vector as per following equation. This is called correction step for state estimate.

$$\hat{X}_r^+(k) = \hat{X}_r^-(k) + K_k[Y_k - h(\hat{X}_r^-(k))] \tag{2.30}$$

Update the covariance with linear covariance dynamics. This is also called correction step for state error covariance.

$$P_k^+ = (I - K_k C_k) P_k^- (I - K_k C_k)^T + K_k R_k K_k^T \tag{2.31}$$

This updated camera state and covariance are used as input in the next iteration and thus extended kalman filter estimates the camera states within a reasonable tolerance bounds. However Q and R are tuning parameters and therefore precaution should be taken while selecting the entries in these matrices. EKF pre-run is carried out before its actual application. It is required to stabilize the initial error and during this period the system dynamics is allowed to propagate without any command.

2.3 Unscented Kalman Filter

Extended kalman filter(EKF) described in previous section is used for recursive nonlinear estimation. EKF however, provides only first order approximation to nonlinear estimation. Another filter known as Unscented kalman filter(UKF) has better performance than EKF. In EKF state distribution is approximated by a Gaussian random variable and the state is propagated through first order approximation of the nonlinear system. However, for UKF the state distribution is approximated by a gaussian random variable, but state distribution is represented using a minimal set of sample points. These sample points capture true mean and covariance of the gaussian random variable. In UKF approach these sample points are propagated through true nonlinear system and capture posterior mean and covariance accurately up to second order.

2.3.1 Unscented transformation

It is difficult to transform a probability density function(pdf) through a nonlinear function. Unscented transformation is a method of calculating statistics of random variable which undergoes a nonlinear transformation. Unscented kalman filter(UKF) philosophy is based on unscented transformation. It performs a nonlinear transformation on a single point rather than a pdf as well as it provides a set of individual points in state space whose sample probability density function(pdf) approximates the true pdf of a state vector.

If the mean and covariance of a state vector x be $\bar{x}$ and P respectively. Then a set of deterministic points called as sigma points can be computed whose ensemble mean and covariance are equal to $\bar{x}$ and P. The dimension of the state vector x is n. Therefore, we choose $2n$ sigma points. The unscented transformation can be expressed as follows

$$\begin{aligned} x^{(i)} &= \bar{x} + \tilde{x}^{(i)} && i = 1, 2, ...2n && (2.32)\\ \tilde{x}^{(i)} &= \left(\sqrt{nP}\right)_i^T && i = 1, 2, ...n && (2.33)\\ \tilde{x}^{(n+i)} &= -\left(\sqrt{nP}\right)_i^T && i = 1, 2, ...n && (2.34) \end{aligned}$$

In the Eq.(2.33)-Eq.(2.34),$\sqrt{nP}$ can be computed by Cholesky decomposition. In camera sensor model each camera has three states r, θ and ϕ therefore the value of n is 3 in the present work carried out for state estimation.

2.3.2 Time update equations

Pin-hole camera vision sensor's n state nonlinear system can be expressed as

$$x_{k+1} = f(x_k, u_k, t_k) + w_k \tag{2.35}$$

$$y_k = h(x_k, t_k) + v_k \tag{2.36}$$

$$w_k \sim (0, Q_k) \tag{2.37}$$

$$v_k \sim (0, R_k) \tag{2.38}$$

where $f(x_k, u_k, t_k)$ is nonlinear transition matrix function as described in Eq.(2.15) and $h(x_k, t_k)$ is nonlinear measurement matrix function as per definition but in present work it is considered linear as described in Eq.(2.28). Q and R are process noise covariance and measurement noise covariance respectively.

The stereovision camera sensor state's mean and covariance can be initialized as per Eq.(2.39)-Eq.(2.40). It is normally assumed that it is known priori or if it is unknown then there is precisely a bound within it is supposed to lie based on the physical sensor measurement precision characteristics.

$$\hat{x}_0^+ = E(x_0) \tag{2.39}$$

$$P_0^+ = E[(x - \hat{x}_0^+)(x - \hat{x}_0^+)^T] \tag{2.40}$$

Time update equations propagate the camera sensor state estimate based on sensor nonlinear dynamics and the averaging of all sigma points state estimates, and propagate camera sensor's error covariance based on averaging the difference between sigma point estimates and its average. Time update equations Eq.(2.41)-Eq.(2.43) for state propagation can be written considering previous step mean and covariance as per Eq.(2.39)-Eq.(2.40) as best guess value for computing next step state mean and error covariance.

$$\hat{x}_{k-1}^{(i)} = \hat{x}_{k-1}^{+} + \tilde{x}^{(i)} \qquad i = 1, 2, ...2n \tag{2.41}$$

$$\tilde{x}^{(i)} = \left(\sqrt{nP_{k-1}^{+}}\right)_i^T \qquad i = 1, 2, ...n \tag{2.42}$$

$$\tilde{x}^{(n+i)} = -\left(\sqrt{nP_{k-1}^{+}}\right)_i^T \qquad i = 1, 2, ...n \tag{2.43}$$

Using the camera sensor nonlinear dynamics, transform the sigma points obtained in Eq.(2.41)-Eq.(2.43) as

$$\hat{x}_k^{(i)} = f(\hat{x}_{k-1}^{(i)}, u_k, t_k) \tag{2.44}$$

Transformed sigma points obtained through nonlinear dynamics as per Eq.(2.44) are again averaged and hence, the prior state estimate at time k is obtained as follows

$$\hat{x}_k^{-} = \frac{1}{2n}\sum \hat{x}_k^{(i)} \tag{2.45}$$

Similarly, prior error covariance taking process noise into account can be propagated as follows.

$$P_k^{-} = \frac{1}{2n}\sum(\hat{x}_k^{(i)} - \hat{x}_k^{-})(\hat{x}_k^{(i)} - \hat{x}_k^{-})^T + Q_{k-1} \tag{2.46}$$

2.3.3 Measurement update equations

Compute the sigma points again with current best guess, which is the prior mean and covariance computed in the previous steps as per Eq.(2.44)-Eq.(2.46), the following step is carried out to enhance the performance of Unscented kalman filter.

$$\hat{x}_k^{(i)} = \hat{x}_k^{-} + \tilde{x}^{(i)} \qquad i = 1, 2, ...2n \tag{2.47}$$

$$\tilde{x}^{(i)} = \left(\sqrt{nP_k^{-}}\right)_i^T \qquad i = 1, 2, ...n \tag{2.48}$$

$$\tilde{x}^{(n+i)} = -\left(\sqrt{nP_k^{-}}\right)_i^T \qquad i = 1, 2, ...n \tag{2.49}$$

Camera sensor measurement model is used to transform the sigma points in previous step as per Eq.(2.47), into predicted measurements. The measurement equation is as follows.

$$\hat{y}_k^{(i)} = h(\hat{x}_k^{(i)}, t_k) \tag{2.50}$$

Taking average of all predicted measurements obtained as per Eq.(2.50), which are corresponding to the sigma points, the predicted measurement output can be computed as follows,

$$\hat{y}_k = \frac{1}{2n}\sum \hat{y}_k^{(i)} \tag{2.51}$$

The covariance of predicted measurement can be computed with $\hat{y}_k^{(i)}$ from Eq.(2.50) and $\hat{y}_k$ from Eq.(2.51) as follows,

$$P_y = \frac{1}{2n}\sum (\hat{y}_k^{(i)} - \hat{y}_k)(\hat{y}_k^{(i)} - \hat{y}_k)^T + R_k \tag{2.52}$$

which is based on difference between predicted measurement and its average.

Similarly, cross covariance between the difference between $\hat{x}_k^{(i)}$ from Eq.(2.44) and prior estimate of camera state $\hat{x}_k^-$ from Eq.(2.45) and the difference between $\hat{y}_k^{(i)}$ from Eq.(2.50) and predicted measurement output $\hat{y}_k$ from Eq.(2.51)can be computed as

$$P_{xy} = \frac{1}{2n}\sum (\hat{x}_k^{(i)} - \hat{x}_k^-)(\hat{y}_k^{(i)} - \hat{y}_k)^T \tag{2.53}$$

The measurement update equation involving kalman gain computation is provide in Eq.(2.54), the required inputs P_y and P_{xy} for kalman gain computation are obtained from previous steps. The measurement update equation for posterior camera state or the corrected state estimate is provided in Eq.(2.55), the required inputs are obtained from previous steps. The measurement update equation for posterior covariance computation is provided in Eq.(2.56), the required inputs are obtained from previous steps. The posterior state estimate and posterior covariance obtained in the following equations are again utilized as the best guess value for next step iteration.

$$K_k = P_{xy} P_y^{-1} \tag{2.54}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(y_k - \hat{y}_k) \tag{2.55}$$

$$\hat{P}_k^+ = \hat{P}_k^- - K_k P_y K_k^T \tag{2.56}$$

2.4 Velocity Estimation

Reactive collision avoidance for moving obstacles involves estimation of obstacle position as well as obstacle velocity [7]. The projection of the moving obstacle on two camera image planes in the geometrical configuration shown in Fig. 2.1 provides the instantaneous position of the obstacle. However, the iterative projections of the moving obstacle acquired by the camera sensors provide the iterative positions of the obstacle and using the difference equation for iterative positions and corresponding time instants the velocity of the obstacle can be computed. The computed velocity serves as a reference or measured velocity.

The camera sensors are noisy and hence, the position and velocity computed by the trianglization method is inaccurate. Therefore, EKF or UKF techniques are used for estimating the accurate position and velocity of the obstacle from the inaccurate data. The camera sensor dynamics is first order differential equation. The sensor dynamics is propagated and the updated position information is obtained. Similarly, differentiate the sensor dynamics to obtain second order state derivative and propagate the second order sensor dynamics and obtain the velocity information. It is assumed that the function is differentiable at all instants of interest. Whenever, obstacle is ahead of camera and visible through both the sensors.

The guidance command uses the estimated obstacle position as well as estimated obstacle velocity to compute the aiming point. The prediction logic decides in advance that what could be the point of intersection or what could be minimum distance between UAV and the moving obstacle. The minimum distance is computed and it is compared with the safety boundary limit of the obstacle if the minimum distance is less than the safe distance then the guidance command is applied to divert the velocity vector of the UAV so that the upcoming collision can be averted. The Fig.2.2 shows the miss distance between UAV and the moving obstacle.

2.4.1 Relative Velocity Estimation

Since the left and right cameras mounted on UAV receive the projection of the obstacle on their image planes if the obstacle lies in the field of view (FoV) of the cameras. The geometrical configuration discussed in the previous section. The initial guess for position and velocity of the obstacle can be obtained from the sequential data frames obtained from stereo camera sensors. Since UAV's own position and velocity is known. The vision sensors provide the relative position and relative velocity estimate of the obstacle. The absolute position and velocity of the obstacle can be estimated.

After initialization, the position as well as velocity of the obstacle is propagated using sensor dynamics as well as the dynamics obtained by taking the derivative of the sensor dynamics respectively. The assumption made here is that obstacle is moving with constant velocity for simplicity. The relative position between UAV and the obstacle can be written as

$$X_{rel} = \begin{bmatrix} x_{rel} & y_{rel} & z_{rel} \end{bmatrix}^T \tag{2.57}$$

The above equation can be represented for UAV in polar coordinates considering the equation in inertial frame. Where r_{rel} , θ_{rel} and ϕ_{rel} are the relative distance and angles between UAV and obstacle. The expression for $\mathbf{r}_{rel}$ is as follows.

$$\mathbf{r}_{rel} = \begin{bmatrix} r_{rel}\cos(\phi_{rel})\cos(\theta_{rel}) & r_{rel}\cos(\phi_{rel})\sin(\theta_{rel}) & -r_{rel}\sin(\phi_{rel}) \end{bmatrix}^T \tag{2.58}$$

The Eq.(2.57) can be written for UAV in polar coordinates considering the equation in velocity frame. Where r_{rel} , χ_{rel} and γ_{rel} are the relative distance and angles between UAV Velocity and axes system. The velocity frame representation is as follows.

$$\mathbf{r}_{rel} = \begin{bmatrix} r_{rel}\cos(\gamma_{rel})\cos(\chi_{rel}) & r_{rel}\cos(\gamma_{rel})\sin(\chi_{rel}) & -r_{rel}\sin(\gamma_{rel}) \end{bmatrix}^T \tag{2.59}$$

Taking first derivative of the Eq.(2.59) the relative velocity between UAV and the obstacle can be written as [1]

$$\begin{bmatrix} \tilde{V}_{xrel} \\ \tilde{V}_{yrel} \\ \tilde{V}_{zrel} \end{bmatrix} = \begin{bmatrix} \cos\gamma_{rel}\cos\chi_{rel} & -r\cos\gamma_{rel}\sin\chi_{rel} & -r\sin\gamma_{rel}\cos\chi_{rel} \\ \cos\gamma_{rel}\sin\chi_{rel} & r\cos\gamma_{rel}\cos\chi_{rel} & -r\sin\gamma_{rel}\sin\chi_{rel} \\ -\sin\gamma_{rel} & 0 & -r\cos\gamma_{rel} \end{bmatrix} \begin{bmatrix} \dot{r}_{rel} \\ \dot{\chi}_{rel} \\ \dot{\gamma}_{rel} \end{bmatrix} \tag{2.60}$$

The absolute velocity of the obstacle can be computed by adding the relative velocity with UAV velocity. UAV's velocity and acceleration information is known through it's dynamics and guidance commands. UAV's position as well as velocity information is also known through it's own predefined global path planning or through the onboard computation as in local path planning. The absolute velocity of the obstacle can be computed through the following equation

$$V_{abs} = V_{uav} + \tilde{V}_{rel} \tag{2.61}$$

The steps followed in velocity estimation is first the camera sensor dynamics is providing the expected sensor's states $\dot{r}, \dot{\theta}, \dot{\phi}$ then with UAV's initial Velocity vector, initial γ and initial χ , it is possible to propagate the UAV system dynamics which is considered as Point mass model and obtain $\dot{V}, \dot{\chi}, \dot{\gamma}$ as well as $\dot{x}, \dot{y}, \dot{z}$ after the state propagation the UAV location is updated and Sensor measures the the new states and again sensor dynamics propagates and the next sensor states are obtained.

Using the obstacle projection on pin-hole camera sensor which are obtained from image processor at each iteration, the relative position of the obstacle can be computed. Using the successive position at each time step, the difference equation is used to compute the velocity of the obstacle. This velocity can be inferred or termed as the measured velocity.

The estimated velocity can be computed using the following steps.

(1). Using the sensor states and their derivatives compute the relative velocity between obstacle and UAV as per Eq.(2.60)

(2). Take the second derivative of sensor dynamics and compute the second order derivative of the sensor states.

$$\begin{bmatrix} \ddot{r} \\ \ddot{\theta} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} -\cos\theta\sin\phi\dot{\phi}V_{xrel} - \sin\theta\cos\phi\dot{\theta}V_{xrel} + \cos\theta\cos\phi\dot{V}_{xrel} \\ -\sin\theta\sin\phi\dot{\phi}V_{yrel} + \cos\theta\cos\phi\dot{\theta}V_{yrel} + \sin\theta\cos\phi\dot{V}_{yrel} \\ +V_{zrel}\cos\phi\dot{\phi} + \dot{V}_{zrel}\sin\phi \\ \frac{r\cos\phi(-\cos\theta\dot{\theta}V_{xrel} - \sin\theta\dot{V}_{xrel} - V_{yrel}\sin\theta\dot{\theta} + \cos\theta\dot{V}_{yrel})}{r^2\cos^2\phi} \\ + \frac{-(-\sin\theta V_{xrel} + \cos\theta V_{yrel})(-r\sin\phi\dot{\phi} + \cos\phi\dot{r})}{r^2\cos^2\phi} \\ \frac{r(-V_{xrel}(\cos\theta\cos\phi\dot{\phi} - \sin\phi\sin\theta\dot{\theta}) - \cos\theta\sin\phi\dot{V}_{xrel}) + (V_{xrel}\cos\theta\sin\phi)\dot{r}}{r^2} \\ + \frac{r(-V_{yrel}(\sin\theta\cos\phi\dot{\phi} + \sin\phi\cos\theta\dot{\theta}) - \sin\theta\sin\phi\dot{V}_{yrel}) + (V_{yrel}\sin\theta\sin\phi)\dot{r}}{r^2} \\ + \frac{r(-V_{zrel}\sin\phi\dot{\phi} + \cos\phi\dot{V}_{zrel}) - V_{zrel}\cos\phi\dot{r}}{r^2} \end{bmatrix} \tag{2.62}$$

Sensor state second order derivatives as obtained in Eq.(2.62) are function of sensor states and their first order derivatives as described in Eq.(2.15), relative velocity as per Eq.(2.60) as well as derivative of relative velocity as described in Eq.(2.66)- Eq.(2.68).

Since, Obstacle velocity is assumed to be constant, the derivative of the obstacle velocity is zero. The UAV velocity is given by the Eq.(A.1-A.3) then by computing the derivative of

UAV velocity the derivative of relative velocity can be computed.

$$V_{rel} = V_o - V_u \tag{2.63}$$

Taking first derivative of Eq.(2.63)

$$\dot{V}_{rel} = \dot{V}_o - \dot{V}_u \tag{2.64}$$

Since obstacle velocity is constant Eq.(2.64) can be written as

$$\dot{V}_{rel} = -\dot{V}_u \tag{2.65}$$

The equations for UAV velocity V_u in inertial frame is provided in the Eq.(A.1-A.3), therefore for $\dot{V}_u$ the Eq.(2.65) can be written as

$$\begin{aligned}
\dot{V}_{xrel} &= V_T \cos(\chi)\sin(\gamma)\dot{\gamma} + \cos(\gamma)\sin(\chi)\dot{\chi} - \cos(\chi)\sin(\gamma)\dot{V_T} && (2.66)\\
\dot{V}_{yrel} &= V_T \sin(\chi)\sin(\gamma)\dot{\gamma} - \cos(\gamma)\cos(\chi)\dot{\chi} - \sin(\chi)\cos(\gamma)\dot{V_T} && (2.67)\\
\dot{V}_{zrel} &= -V_T \cos(\gamma)\dot{\gamma} - \sin(\gamma)\dot{V_T} && (2.68)
\end{aligned}$$

(3). Propagate the sensor state derivative and using the state derivative compute the propagated the relative velocity. The obstacle velocity is computed from the data received from the camera sensors using difference equation as follows

$$V_{cam} = \frac{X_{obs}(t+dt) - X_{obs}(t)}{dt} \tag{2.69}$$

The obstacle Velocity obtained through Eq.(2.69) is assumed as obstacle velocity through camera sensor measurement as obstacle positions are computed by triangulization method at each instant. However, the obstacle velocity obtained by Eq.(2.61), which involves relative velocity computation and is achieved through sensor dynamics propagation. The EKF or UKF filter is used to compute the estimated absolute velocity of the obstacle as follows.

$$V_{est}(k+1) = V_{est}(k) + K_v(V_{abs} - V_{cam}) \tag{2.70}$$

where,K_v is kalman filter gain.

Since, the UAV velocity is known and relative velocity between UAV and obstacle is known the absolute velocity of obstacle can be computed.

2.4.2 Reactive collision avoidance with moving obstacles

UAV and the obstacle's initial position is $X_u(0)$ and $X_o(0)$ and their respective velocity is V_u and V_o. UAV is moving at constant velocity as no guidance is working and obstacle is assumed to be moving with constant velocity as per Fig. 2.2. The predicted positions $X_u(t)$ and $X_o(t)$ of the UAV and the obstacle with current velocities after time t is given by Eq.(2.71)-Eq.(2.72).The prediction logic tells the future positions of UAV and obstacle and also the minimum separation between UAV and obstacle. Therefore, it is possible for UAV to take corrective action in due course of time and avoid any future collision by applying the guidance command and changing it's velocity profile.

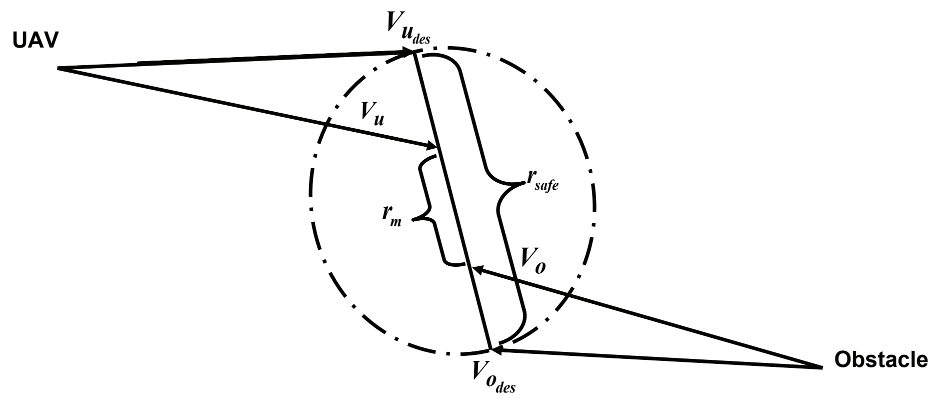


Figure 2.2: Non-cooperative conflict resolution

$$X_u(t) = X_u(0) + t * V_u \tag{2.71}$$
$$X_o(t) = X_o(0) + t * V_o \tag{2.72}$$

Relative distance predicted between UAV and obstacle can be computed at any time t as per Eq.(2.73), which is a vector equation. The Euclidian norm of $R_{rel}(t)$ provides the predicted distance $d(t)$ in scalar form and discussed later in the section.

$$R_{rel}(t) = X_u(t) - X_o(t) \tag{2.73}$$

Subtracting Eq.(2.72) from Eq.(2.71) and substituting with Eq.(2.75) and Eq.(2.73), the following equation can be obtained.

$$R_{rel}(t) \quad = \quad R_{rel}(0) + t(V_u - V_o) \tag{2.74}$$

Where,Eq.(2.75) describes the initial separation between UAV and obstacle.

$$R_{rel}(0) \quad = \quad X_u(0) - X_o(0) \tag{2.75}$$

The objective here is to compute the minimum separation between the UAV and the obstacle and the corresponding predicted time. The predicted point when the UAV just intersect each other or at a minimum distance to each other the minimum Distance $d(t)$ should be Minimum of $\|R_{rel}(t)\|$. In order to compute the minimum distance

$$\begin{aligned}
D(t) &= d(t)^2 = \|R_{rel}(t)\|^2 && (2.76)\\
&= R_{rel}(t)^T R_{rel}(t) && (2.77)\\
&= (R_{rel}(0) + tV_u - tV_o)^T (R_{rel}(0) + tV_u - tV_o) && (2.78)\\
D(t) &= (R_{rel}(0))^T (R_{rel}(0)) + 2(R_{rel}(0))^T (V_u - V_o)t + (V_u - V_o)^T (V_u - V_o)t^2 && (2.79)
\end{aligned}$$

Differentiating the Eq.(2.79) for $D(t)$ with respect to time t and solving for minimum distance we obtain as follows.

$$\frac{dD(t)}{dt} \quad = \quad 2(V_u - V_o)^T (V_u - V_o)t + 2(R_{rel}(0))^T (V_u - V_o) \tag{2.80}$$

For $D(t)$ to be minimum,

$$\frac{dD(t)}{dt} \quad = \quad 0 \tag{2.81}$$

Solving the Eq.(2.81) for t, which is called as "time of closest approach" t_c when the UAV and the obstacle are closer to each other as per Eq.(2.82). t_c is the available time for UAV to maneuver and avoid the predicted collision depending on the predicted minimum separation.

$$t_c = -\frac{R_{rel}(0)^T (V_u - V_o)}{\|V_u - V_o\|^2} \tag{2.82}$$

The distance between UAV and obstacle at the time of closest approach is called as miss distance or zero effort miss distance the ZEM can be computed as

Substituting the value of t_c from Eq.(2.82) in Eq.(2.83) the predicted miss distance r_m is computed as

$$r_m = X_u(t_c) - X_o(t_c) \tag{2.83}$$

Where $X_u(t_c)$ and $X_o(t_c)$ are the predicted position of UAV and obstacle at time t_c for predicted reactive collision to occur the folowing conditions must be satisfied (i) If $t_c > 0$ the UAV and obstacle are coming closer to each other therfore the probable conflict is possible. If $t_c < 0$, then UAV and obstacle are moving away from each other. (ii) If $\|r_m\| < r_{safe}$ then predicted position of UAV is within the safety sphere of obstacle and it can be inferred that collision has occured. Where, r_{safe} is the safety radius of the sphere. If both conditions are satisfied simultaneously then the collision is said to be occurred at the time of closest approach t_c. Therefore, the effective guidance commands must be computed to avert the collision. In non-cooperative scenario where obstacle is autonomously moving the UAV has to maneuver to avoid collision. The UAV has to deflect from its predicted position by residual distance r_{res} which is given in Eq.(2.84) so that the r_{safe} separation can be insured.

$$r_{res} = r_{safe} - \|r_m\| \tag{2.84}$$

The Eq.(2.84) can be incorporated by changing the velocity vector of the UAV and the desired velocity of UAV $V_{u_{des}}$ is provided in Eq.(2.85). The UAV velocity vector need to align along the desired velocity so that the desired aiming point can be achieved.

$$V_{u_{des}} = \frac{V_u t_c + r_{vm}}{\|V_u t_c + r_{vm}\|} \tag{2.85}$$

where, r_{vm} is defined in the Eq.(2.86), it is a distance vector with magnitude r_{res} and direction r_m.

$$r_{vm} = \frac{r_{res} r_m}{\|r_m\|} \tag{2.86}$$

Therefore, the differential geometric guidance(DGG) can be invoked to demand the desired velocity Eq.(2.85) and correct the predicted position of the UAV to avoid collision with obstacle.

Chapter 3

Collision Avoidance Philosophy and Guidance Design

In the present work, it is assumed that global path planning has already been done off line (or possibly loaded to the onboard processor through telemetry), where a path towards the goal point has already been found accounting for the known obstacles in the environment. The UAV is equipped with an onboard camera with necessary image processing algorithms to sense an unforseen obstacle, along with its location, in its close vicinity. The main focus then is to predict a possible collision with the obstacle, and if so, to steer it away with appropriate generation of guidance commands. Details of this guidance algorithm are discussed in this section.

3.1 Aiming Point Computation

Once the obstacle information is available through onboard sensors, the task is to predict possible collision, and if so, to steer it away by appropriately computing of aiming point and generation of required guidance commands. The philosophy behind computing the aiming point is based on collision cone approach has physical relevance with torch light experiment, where if the torch light is glown to the sphere, the locus of extreme points of the glow area will form a circle on the sphere and the rays passing through the extreme points are tangent to the sphere. The same analogy is replicated in 3D representation of collision cone containing UAV's CG at coordinate frame origin, point obstacle with safety sphere, plane and a circle as shown in Fig. 3.1. Note that if one draws tangents to this sphere from the CG of the UAV, it results in a 'cone' (hence it is called as the 'collision cone' [41]), and the circle formed by locus of tangents contact point on sphere is named as collision circle and represented as C^{η}

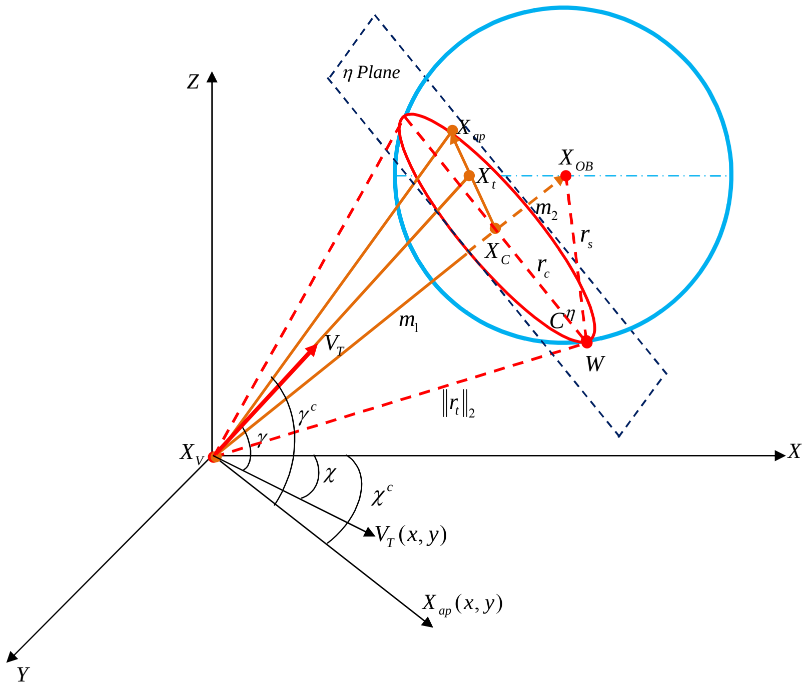


Figure 3.1: Collision Cone Representation

The coordinate frame assumed in Fig. 3.1 is parallel to the inertial frame and located on the centre of gravity (CG) of the UAV. Let the UAV fly along the direction of the V_T. It is also assumed that the angle of attack and sideslip angles are small so that vehicle body X-axis (along which the onboard sensor is installed) can detect obstacles along the V_T direction. Once an obstacle is detected by onboard sensors, collision avoidance algorithm is invoked and a new aiming point is computed. As per collision cone approach the computed aiming point is tangent to the sphere, so it should lie on this collision circle. The centre and radius of the collision circle is determined by using the information of obstacle sphere radius and distance between UAV and centre of the obstacle. From Fig. 3.1, let us consider the $\Delta X_V W X_{OB}$. Where, X_V is the UAV position, W is a point on the collision circle and X_{OB} is the obstacle position. Then by applying Pythagoras theorem on $\Delta X_V W X_{OB}$

$$\|X_r\|^2 = \|r_t\|^2 + r_s^2 \tag{3.1}$$

$$\|r_t\|^2 = X_r^2 - r_s^2 \tag{3.2}$$

$$\|X_r\| = m_1 + m_2 \tag{3.3}$$

where r_s is the radius of the safety sphere, r_t is the tangent vector joining UAV position X_V and W, the line-of-sight (LOS) X_r, by definition, is the line joining the CG of the UAV to the centre of the obstacle. X_c is the centre of the collision circle and the Euclidean distance between this centre to X_V is m_1 and to X_{OB} is m_2 respectively. Here m_1 and m_2 are also the components of the line segment $\|X_r\|$. Applying Pythagoras theorem on $\Delta X_V W X_C$.

$$\begin{aligned} \|r_t\|^2 &= m_1^2 + r_c^2 \\ r_c^2 &= \|r_t\|^2 - m_1^2 \end{aligned} \tag{3.4}$$

Where r_c is the radius of the collision circle, from $\Delta X_V W X_{OB}$

$$r_s^2 = m_2^2 + r_c^2 \tag{3.5}$$

$$\begin{aligned} m_2^2 &= r_s^2 - r_c^2 \\ &= r_s^2 - \|r_t\|^2 + m_1^2 \end{aligned} \tag{3.6}$$

To find the value of m_1 and m_2, Squaring Eq.(3.3) on both sides and substitute Eq.(3.6), yields

$$\|X_r\|^2 = (m_1 + m_2)^2 = m_1^2 + m_2^2 + 2m_1 m_2 \tag{3.7}$$

$$\begin{aligned} m_1 &= \frac{\|X_r\|^2 - r_s^2 + \|r_t\|^2}{2\|D_r\|} \\ &= \frac{\|r_t\|^2}{\|D_r\|} \qquad (3.8) \\ m_2 &= \frac{r_s^2}{\|D_r\|} \qquad (3.9) \end{aligned}$$

The m_2 value can be find out by substituting the value of m_1 in Eq. (3.3)

$$\begin{aligned} r_c^2 &= \|r_t\|^2 - m_1^2 \\ r_c &= \frac{\|r_t\| r_s}{D_r} \qquad (3.10) \end{aligned}$$

The centre of the collision circle is given as

$$X_C = X_V + \frac{m_1}{D_r}(X_{OB} - X_V) \tag{3.11}$$

Let us consider the plane η, which contains the collision circle C_η. To find the intersection point between the plane η and velocity vector V_T projected in time t from the UAV's current position. Due to geometrical symmetry, the vector joining collision circle centre (x_c, y_c, z_c) and obstacle centre (x_{ob}, y_{ob}, z_{ob}) is always perpendicular to plane η . Therefore, the normal vector $\hat{n}$ to the plane can be described as

$$\hat{n} = (a, b, c) = (x_{ob} - x_c, y_{ob} - y_c, z_{ob} - z_c) \tag{3.12}$$

The general plane equation passing through a point (x_c, y_c, z_c) is

$$a(x - x_c) + b(y - y_c) + c(z - z_c) = 0 \tag{3.13}$$

The position vector of UAV is expressed in terms of instantaneous velocity vector projected in time t is expressed as

$$X_t = X_V + tV \tag{3.14}$$

Substitute above equation into plane equation and solve for desired time t^*, which is termed as time to reach the plane

$$a(x_v + tu - x_c) + b(y_v + tv - y_c) + c(z_v + tw - z_c) = 0 \tag{3.15}$$

$$t^* = -\frac{\hat{n} \cdot (X_V - X_C)}{\hat{n} \cdot V} \tag{3.16}$$

The intersection point is given as

$$X_{t^*} = X_V + t^* V \tag{3.17}$$

Once intersection point is computed, the following conflict conditions are evaluated

1. $\|X_{t^*} - X_C\| <$ Radius of the circle

2. $t^* > 0$

If the above conditions are satisfied then the obstacle under consideration is said to be critical, then the new aiming point is computed as follows

$$X_{ap} = X_C + r_c \frac{X_{t^*} - X_C}{\|X_{t^*} - X_C\|} \tag{3.18}$$

Note that, when no obstacle is critical, the goal point becomes the aiming point, i.e. $X_{ap} = X_{goal}$.

The collision avoidance problem therefore can be interpreted as a sequential target interception problem or way point guidance problem with angle constraints at intermediate targets or way points. However, the basic difference is the online fixing of these intermediate points and then quickly realigning the velocity vector to visit them this is done by the computation of required guidance commands as follows. Once the aiming point is computed, the task then is to follow the philosophy of 'aiming point guidance' of missile guidance literature [18] (which has a close resemblance to pursuit guidance [33]) and align the vector towards this aiming point. For carrying out the necessary algebra (see Fig. 2), the obstacle coordinates are given by $X_{ob} = [x_{ob} \;\; y_{ob} \;\; z_{ob}]$ and the relative aiming point vector is given by $X_{vap} = [x_{vap} \;\; y_{vap} \;\; z_{vap}]$. Once these co-ordinates are known (with respect to the coordinate frame at CG that is parallel to the intertial frame), the desired flight path angle γ^c and course angle χ^c can be calculated as:

$$\gamma^c = \tan^{-1}\left(\frac{z_{vap}}{\sqrt{x_{vap}^2 + y_{vap}^2}}\right) \tag{3.19}$$

$$\chi^a = \tan^{-1}\left(\frac{y_{vap}}{x_{vap}}\right) \tag{3.20}$$

Note that even though it may sound logical to slow down (i.e. to reduce V_T) until the collision is avoided, in this work it is proposed to hold V_T at its initial value at the start of guidance (i.e. $V_T^c = V_T(0)$). This is because the time availability is small and very less correction can be done for it within the available small time-to-go (usually thrust can be varied only slowly). The angles γ and χ as well as V_T^c are the necessary guidance commands in the formulation using the kinematic model. In Fig. 2, $V_T(x, y)$ is the projection of the velocity vector and $X_{vap}(x, y)$ is the projection of the revised LOS in the horizontal plane respectively. For the formulation using the point mass model, however, one needs to carry out further algebra to generate the physically meaningful guidance commands, i.e. the angle of attack, bank angle and thrust commands. Note that the alignment of velocity vector needs to be quickly carried out within a fraction of the available time-to-go, thereby making it possible to avoid pop-up obstacles. For this to happen, the information regarding time-to-go to the aiming point first need to be known, which is obtained in following manner. Note that V_T needs to be aligned to the vector, which is located at the CG of the vehicle and points towards the aiming point. This vector can be interpreted as 'Revised LOS', the magnitude of which is given by

$$R = \|X_{vap}\|_2 = \sqrt{x_{vap}^2 + y_{vap}^2 + z_{vap}^2} \tag{3.21}$$

Assuming that V_T remains fairly constant (which is true, as our formulation attempts to assure it), the time-to-go to reach the aiming point can be computed as follows.

$$t_{go} = \frac{(X_{vap} \cdot V_T)}{\|V_T\|^2} = \frac{R}{V_T} \tag{3.22}$$

Next, the settling times of the imposed error dynamics are selected as a fraction of this time-to-go and gain values are selected accordingly.

3.2 Nonlinear Differential Geometric Guidance

The present work assumes a goal point in the environment with unforseen obstacles whose instantaneous locations are known through onboard sensors. Aiming point is calculated with the collision cone approach [41]. The aiming point is pursued through nonlinear geometric guidance law. In the present work, the velocity magnitude of UAV is kept constant and only directions are manipulated to execute the guidance law. The objective of the guidance law is to steer the UAV to the aiming point through correction of velocity direction both in the vertical and the horizontal plane. This implies the desired values for the orientation of the velocity vector in both the planes respectively will become

$$\gamma^c = \lambda_e, \quad \chi^c = \lambda_a$$

as evaluated from Eq. (3.19) and Eq. (3.20). To execute the guidance law with the point mass model as described by Eqs. (A.7) - (A.12), the necessary forces are required to stir the vehicle in the desired direction. Forces required by the UAV are observed in form of three control variables–desired angle of attack α_d ,the desired bank angle μ_d and the desired thrust T_d.

3.3 Controller Design

The control commands μ_d and α_d are required to achieve the commanded horizontal and vertical flight path angles whereas T_d is required to track the desired velocity of the vehicle. Desired bank angle μ_d is generated in the closed form as follows:

$$\begin{aligned} mV_T \cos\gamma \dot{\chi} &= (T \sin\alpha + L) \sin\mu && (3.23) \\ mV_T \dot{\gamma} + mg \cos\gamma &= (T \sin\alpha + L) \cos\mu && (3.24) \end{aligned}$$

Desired bank angle μ_d is generated by using the philosophy of NDI [35]. Enforcing the first order error dynamics on $\dot{\chi}$ and $\dot{\gamma}$ we get

$$\begin{aligned} \dot{\chi} &= -k_\chi (\chi - \chi^c) && (3.25) \\ \dot{\gamma} &= -k_\gamma (\gamma - \gamma^c) && (3.26) \end{aligned}$$

Since the bank angle is appearing in Eq. (3.23) and in Eq. (3.24) in affine form, so both the equations are used to find the desired value of the bank angle. By dividing Eq. (3.23) by

Eq. (3.24)we get,

$$tan\mu_d = \frac{mV_T \cos\gamma\dot{\chi}}{mV_T\dot{\gamma} + mg\cos\gamma} \tag{3.27}$$

$$\mu_d = \arctan\frac{mV_T\cos\gamma(-k_\chi(\chi-\chi^c))}{mV_T(-k_\gamma(\gamma-\gamma^c)) + mg\cos\gamma} \tag{3.28}$$

The desired thrust T_d is required by the vehicle to maintain the velocity to its desired value ${V_T}^c$. ${V_T}^c$ is the initial velocity of the vehicle. It is assumed that in the short duration of obstacle avoidance the velocity of UAV remains constant. The first order error dynamics for the velocity can be written as

$$\dot{V}_T = -k_V\left(V_T - {V_T}^c\right) \tag{3.29}$$

It can be seen from Eqs. (A.10), (A.11)and (A.12) that both the thrust and angle of attack (α) are dependent on each other and hence closed form solution for both control variables cannot be deduced by algebraic manipulation. $\dot{V}_T$ dynamics in Eq. (A.12) can be rewritten as

$$T\cos\alpha - D = m\dot{V}_T + mg\sin\gamma \tag{3.30}$$

By substituting Eq. (3.29) in Eq. (3.30) we get,

$$T\cos\alpha - D = m(-k_V V_T - {V_T}^c) + mg\sin\gamma \tag{3.31}$$

Similarly, by squaring and adding equation Eqs. (3.23) and (3.24) we get,

$$(T\sin\alpha + L)^2 = (mV_T\cos\gamma\dot{\chi})^2 + (mV_T\dot{\gamma} + mg\cos\gamma)^2 \tag{3.32}$$

$$(T\sin\alpha + L) = \sqrt{(mV_T\cos\gamma\dot{\chi})^2 + (mV_T\dot{\gamma} + mg\cos\gamma)^2} \tag{3.33}$$

By substituting Eq. (3.25) and Eq. (3.26) in Eq. (3.33) we get,

$$(T\sin\alpha + L) = \sqrt{(mV_T\cos\gamma\dot{(} - k_\chi(\chi-\chi^c)))^2 + (mV_T(-k_\gamma(\gamma-\gamma^c)) + mg\cos\gamma)^2} \tag{3.34}$$

The two Eqs. (3.31) and (3.34) are nonlinear and have two unknowns T_d and α_d. α_d and T_d are generated by solving numerically, simultaneously Eqs. (3.31) and (3.34) through Newton Raphson method [40]. The initial guess values of the control variables for Newton Raphson method are taken as trim values $\alpha = 3.314^0$ and $T = 5.56N$. Learning rate parameter of 0.6 is introduced for both the variables for smooth convergence. The convergence criteria is set as the bounded relative error in T_d and α_d or the maximum number of iteration assigned for convergence. Bounded relative error can be stated as

$$\frac{\Delta\alpha}{\alpha} < Tol_1 \quad \frac{\Delta T}{T} < Tol_2 \tag{3.35}$$

where Tol_1 and Tol_2 are respective tolerance limits of the relative error in T_d and α_d. Tol_1 and Tol_2 can be taken differently but for present study $Tol_1 = Tol_2 = 0.1\%$. The maximum number of iterations assigned for convergence is 26. The time for convergence is around $1ms$ which is lesser than one time update cycle ($20ms$).

3.3.1 Autopilot Compensation

The control variables $\mu^*,\ \alpha^*,\ T^*$ are passed through the autopilot to account for internal dynamics. The autopilot introduces a first order delay in response which is compensated by designing an autopilot controller in all the three control channels. It is assumed that the autopilot states (control variables $\mu_d,\ \alpha_d,\ T_d$) are available for the feedback. The controller is designed based on the error of the actual states of the autopilot $\mu_d,\ \alpha_d,\ T_d$ and the desired state of the autopilot $\mu^*,\ \alpha^*,\ T^*$ respectively. The error in the channel of bank angle is given by $e = \mu_d - \mu^*$. Enforcing the first order error dynamics on the bank angle we get

$$(\dot{\mu_d} - \dot{\mu^*}) + k_{\mu_{dc}}(\mu_d - \mu^*) = 0 \tag{3.36}$$

$$\dot{\mu_d} \;=\; \dot{\mu^*} - k_{\mu_{dc}}(\mu_d - \mu^*) \tag{3.37}$$

Substituting the autopilot dynamics from the Eq. (A.15) in Eq. (3.37) and rearranging we get

$$\mu_{dc} = \frac{1}{k_{\mu_d}}[k_{\mu_d}\mu_d + \dot{\mu^*} - k_{\mu_{dc}}(\mu_d - \mu^*)] \tag{3.38}$$

desired rate of change of control $\dot{\mu^*} = 0$, therefore,

$$\mu_{dc} = \frac{1}{k_{\mu_d}}[k_{\mu_d}\mu_d - k_{\mu_{dc}}(\mu_d - \mu^*)] \tag{3.39}$$

Similar controllers for autopilot compensation are designed for the remaining control variables as $\alpha_{dc},\ T_{dc}$.

3.4 Noise-free Performance of the System

It is mandatory to check the performance of the system in noise-free situations. The robustness of the system can be verified in noisy conditions in the later sections. Both the kinematic model as well as point mass model of UAV are considered for simulation studies. The reactive collision avoidance algorithm is applied for the UAV in a single obstacle scenario and the performance of the algorithm is evaluated.

3.4.1 Kinematic Model Noise-free Performance: Stationary Obstacle

The kinematic model of UAV is considered for reactive collision avoidance in noise-free scenario. The simulation is carried out with stationary single obstacle the Initial Position of UAV is $[0, 0, 0]$, Initial Velocity $[19.9, 0.97, 0.99]$, Target Position $[350, -5, -5]$, Obstacle position $[200, -2.5, -2.5]$ and UAV final location $[345.6, -4.0, -4.0]$. Fig. 3.2 shows the trajectory of the UAV. The guidance commands for the UAV with kinematic model is shown in Fig. 3.3. The guidance for χ and γ are implemented through first order autopilot lag equations. Velocity of the UAV is kept constant throughout the simulation and aiming point is achieved through variation of χ and γ. Therefore, UAV with kinematic model is able to avoid the obstacle in noise-free scenario.

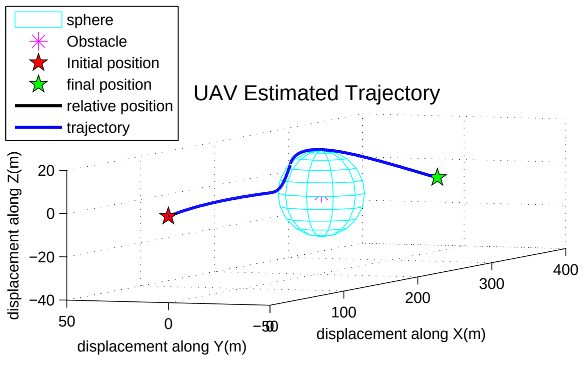


Figure 3.2: UAV Kinematic model with Single Stationary Obstacle: UAV Trajectory

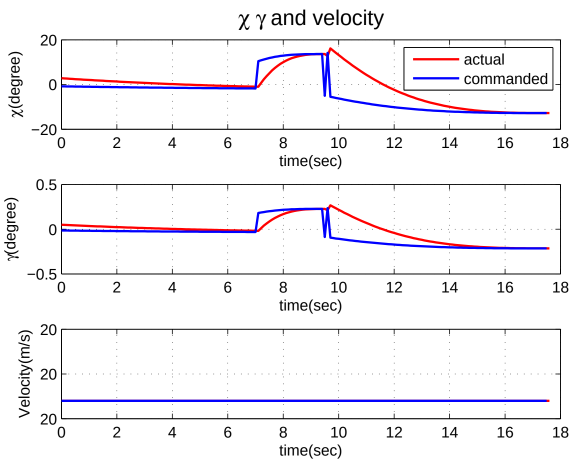


Figure 3.3: UAV Kinematic model with Single Stationary Obstacle: χ, γ and Velocity

3.4.2 Point mass Model Noise-free Performance: Stationary Obstacle

The point mass model of UAV is considered for reactive collision avoidance in noise-free scenario. This model has a close resemblance with practical UAVs as the aerodynamic lift and drag force coefficients are taken into consideration. The simulation is carried out with stationary single obstacle the Initial Position of UAV is $[0, 0, 50]$, Initial Velocity $[19.9, 0.97, 0.99]$, Target Position $[350, -5, 52]$, Obstacle Position $[140, -3, 51]$ and UAV final location $[346.18, -4.35, 52.29]$. Fig. 3.4 shows the trajectory of the UAV. The guidance commands for the UAV with point mass model is shown in Fig. 3.5. The initial values of guidance commands are $\alpha = 2.314^0$, $\mu = 0^0$ and $Thrust = 5N$. The commands are exercised during the collision avoidance and afterwards the commands are almost constant as the destination is determined.

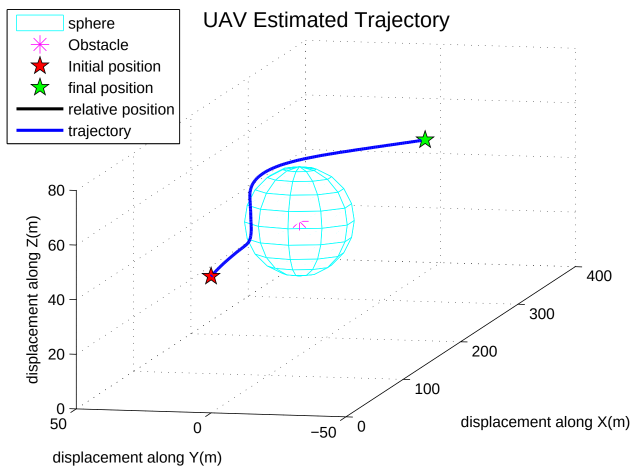


Figure 3.4: UAV Point mass model with Single Stationary Obstacle: UAV Trajectory

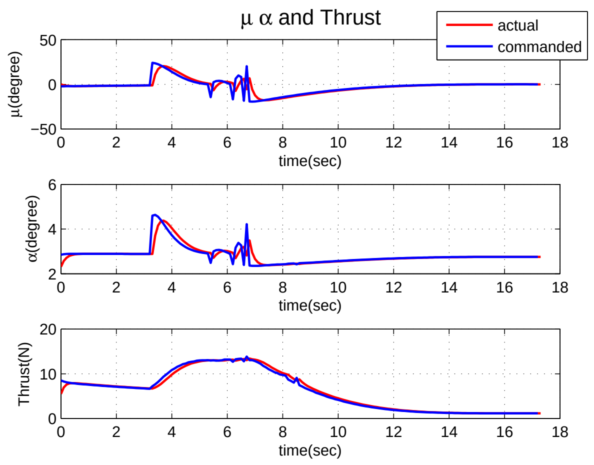


Figure 3.5: UAV Point mass model with Single Stationary Obstacle: μ, α and Thrust

3.4.3 Point mass Model Noise-free Performance: Moving Obstacle

The point mass model of UAV is considered for reactive collision avoidance in noise-free scenario. The simulation is carried out with single moving obstacle with Velocity $[2, 0, 0]$, the Initial position of UAV is $[0, 0, 50]$, Initial Velocity $[19.9, 0.97, 0.99]$, Target Position $[350, -5, 52]$, Obstacle initial position $[140, -3, 51]$, Obstacle final position $[174.8, -3, 51]$

and UAV final location $[346.63, -4.3, 52.3]$. Fig. 3.6 shows the trajectory of the UAV. Since the obstacle is moving, the final location of obstacle is shown in the trajectory. The guidance commands for the UAV with point mass model is shown in Fig. 3.7. The commands are executed to avoid collision with the dynamic obstacle as the aiming point computation is incurred due to relative motion caused by UAV as well as the obstacle motion.

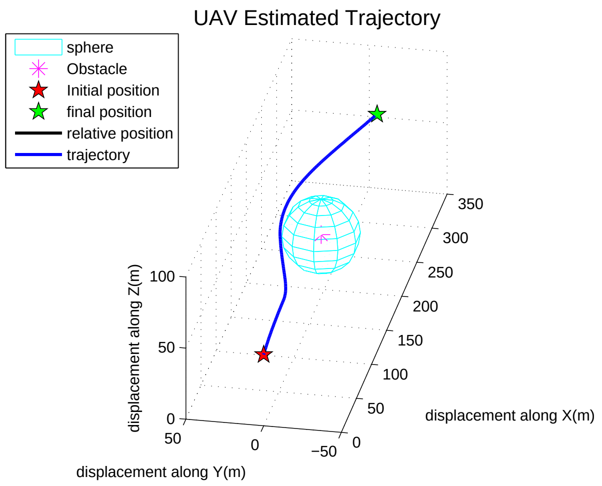


Figure 3.6: UAV Point mass model with Single Moving Obstacle: UAV Trajectory

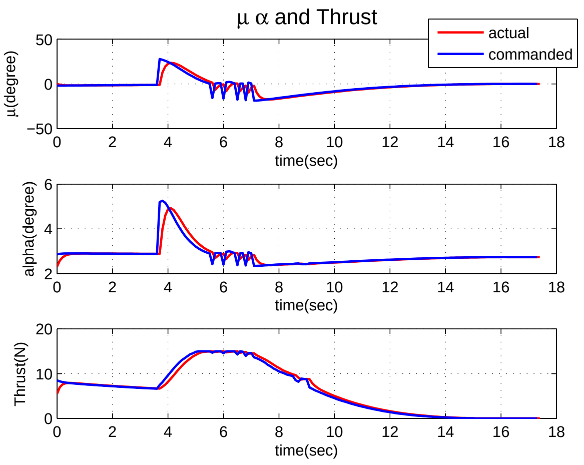


Figure 3.7: UAV Point mass model with Single Moving Obstacle: μ, α and Thrust

Fig. 3.8 shows the obstacle's true and estimated velocity profile. The estimated velocity through first order differential of sensor dynamics is inaccurate and fluctuating. In later sections the obstacle velocity is accurately estimated using extended kalman filter as well as unscented kalman filter. Fig. 3.9 shows the minimum distance profile. The guidance commands for α and χ show some fluctuations at around 6 to 8 seconds which is due to UAV's close proximity with obstacle safety sphere and invoking sphere tracking algorithm [9].

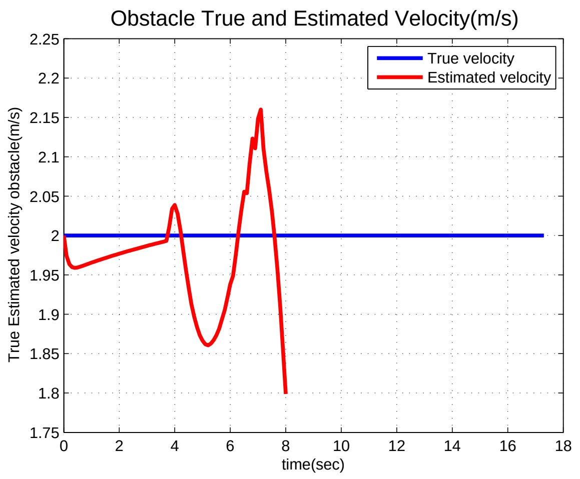


Figure 3.8: UAV Point mass model with Single Moving Obstacle: Velocity Profile

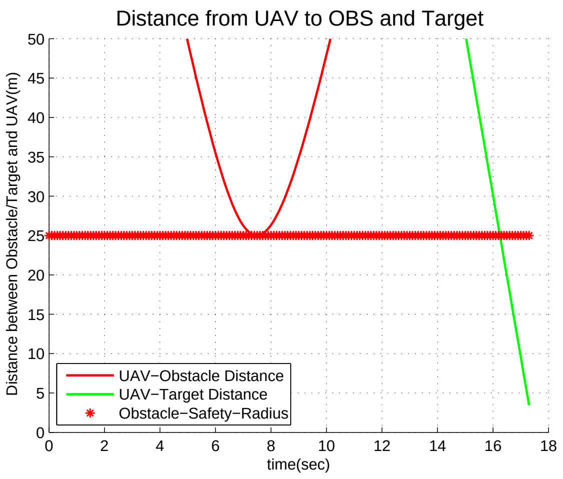


Figure 3.9: UAV Point mass model with Single Moving Obstacle: Minimum Distance Profile

3.5 Extension of the Algorithm for Multiple Stationary Obstacles

Without loss of generality, multiple stationary obstacles are considered here in the problem formulation. The formulation can be treated as 'reconfigurable' [48] in the sense that if some of the obstacles are in close proximity with each other, then one composite obstacle is considered encircling all of these obstacles rather than individual obstacles. The collision avoidance algorithm is implemented considering only the composite obstacles and remaining individual obstacles.

3.5.1 Formulation of Reconfigurable Obstacles

Let considered the onboard sensor mounted on the UAV senses the N obstacles. The information of position and safety sphere radius of each obstacle are obtained through the onboard sensors. First all the N obstacles position and safety radius are assigned in a vector form as X and R_v respectively. One of the obstacles is considered as global obstacle say X_{OG} and the distance between this obstacle centre and remaining other $N-1$ obstacles centres are computed. If any of the $N-1$ obstacles are in close proximity of the global obstacle (i.e. distance between the two obstacle centres is less than the summation of two obstacle safety sphere radius and minimum safety distance considered between them) then one composite safety sphere is formed, which encompassing the safety spheres of both global obstacle and the close proximity obstacle in following manner. Let us assume that i^{th} obstacle in obstacle position vector $X(:,i)$ is in close proximity to global obstacle, then a unit vector along the line joining the centre of the global obstacle and close proximity obstacle is given as.

$$X_{rv} = \frac{X(:,i) - X_{OG}}{\|X(:,i) - X_{OG}\|} \tag{3.40}$$

The radius and centre of composite safety sphere are compute as follows

$$\begin{aligned} r_a &= (r_g + R_v(i) + \|X(:,i) - X_{OG}\|)/2 \\ X_a &= X_{OG} + (r_a - r_g)X_{rv} \end{aligned} \tag{3.41}$$

Where, $R_v(i)$ and r_g are the safety sphere radius of the close proximity and global obstacle respectively. X_a and r_a are the centre and radius of the composite safety sphere respectively.

There may be situation such that either one of the obstacle safety sphere is completely lie inside the other obstacle safety sphere. In such scenario the computed r_a comes either less than or equal to the bigger obstacle safety sphere radius. Therefore, choose the centre and the radius of the bigger obstacle safety sphere as the composite safety sphere. This composite safety sphere is updated as the global obstacle and the process is repeated for remaining obstacles considering one by one steps. The obstacles which are not in close proximity to the considered global obstacle are regarded as separate obstacles. The position and safety radius information of the separate obstacles are stored in a separate vector as X_{sep} and r_{sep} respectively. Further this separate vector X_{sep} along with global obstacle is put together and it forms a new set of obstacles. This method is iterated for this new set in one by one steps considering each one as global obstacle. Finally, the N obstacle set is transformed into a new final set say X_{New} which includes clustered global obstacles as well as un-clustered global obstacles. The entire logic is described in Fig. 3.10 in flowchart format.

3.5.2 Collision Avoidance algorithm for Multiple Individual Obstacles

Initialize the UAV aiming point as a goal point $X_{ap} = X_{goal}$ and corresponding time-to-go as minimum time to reach $t_{min} = t_{go}$ and initialize the final set of obstacles X_{New}. Then compute the aiming point X_{apk}, time to reach the plane t_k^* and check the collision criteria for the k^{th} global obstacle in new set X_{New} using Eq.(3.16) to Eq.(3.18). If the collision criteria is satisfied and computed t_k^* is lesser than t_{min} then update the UAV aiming point as $X_{ap} = X_{apk}$ and the minimum time to reach as $t_{min} = t_k^*$. The above procedure is repeated for remaining all global obstacles in the new set X_{New} one by one. More detailed logic is given in flowchart as shown in Fig. 3.11

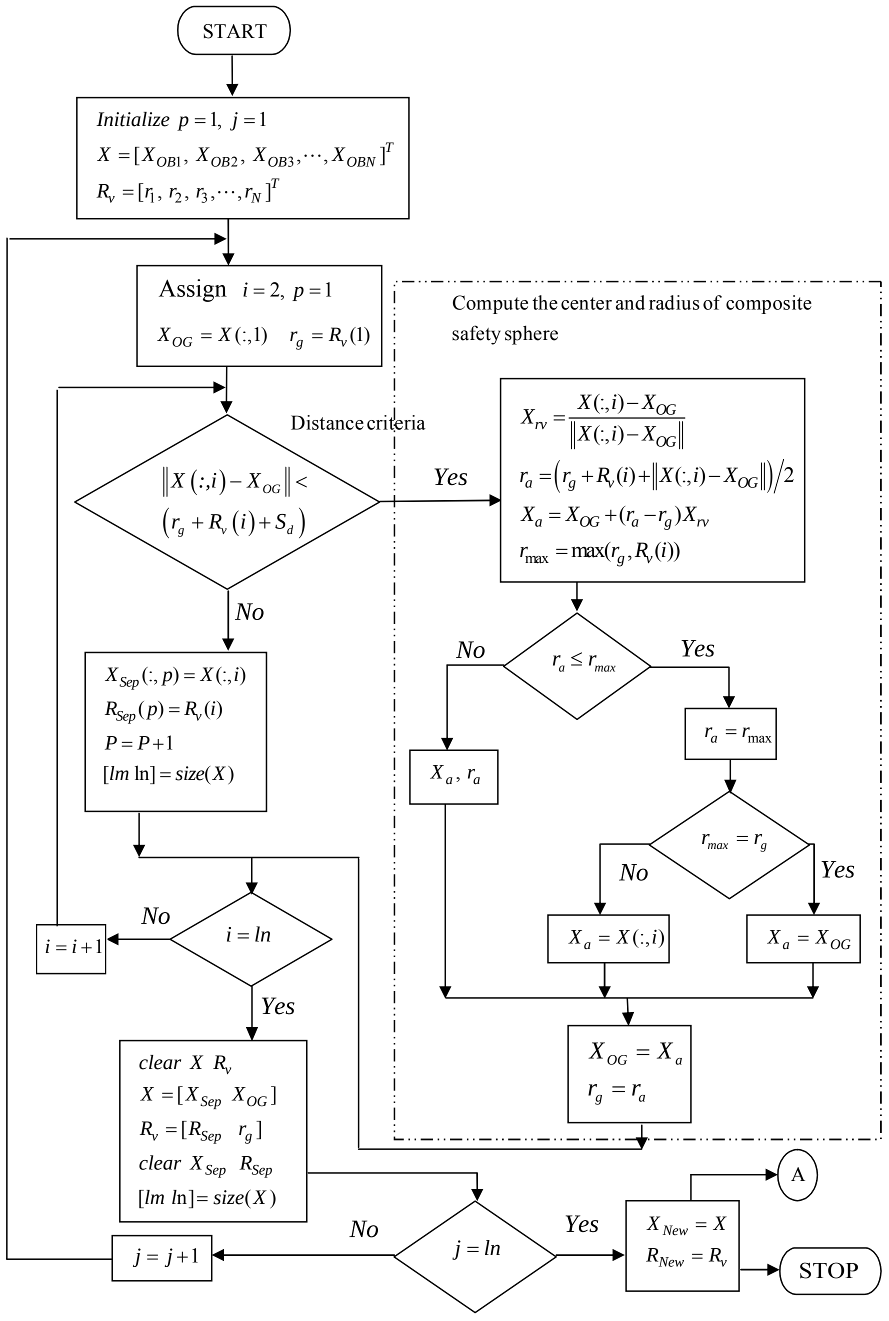


Figure 3.10: Flowchart for formulation of Reconfigurable Obstacles

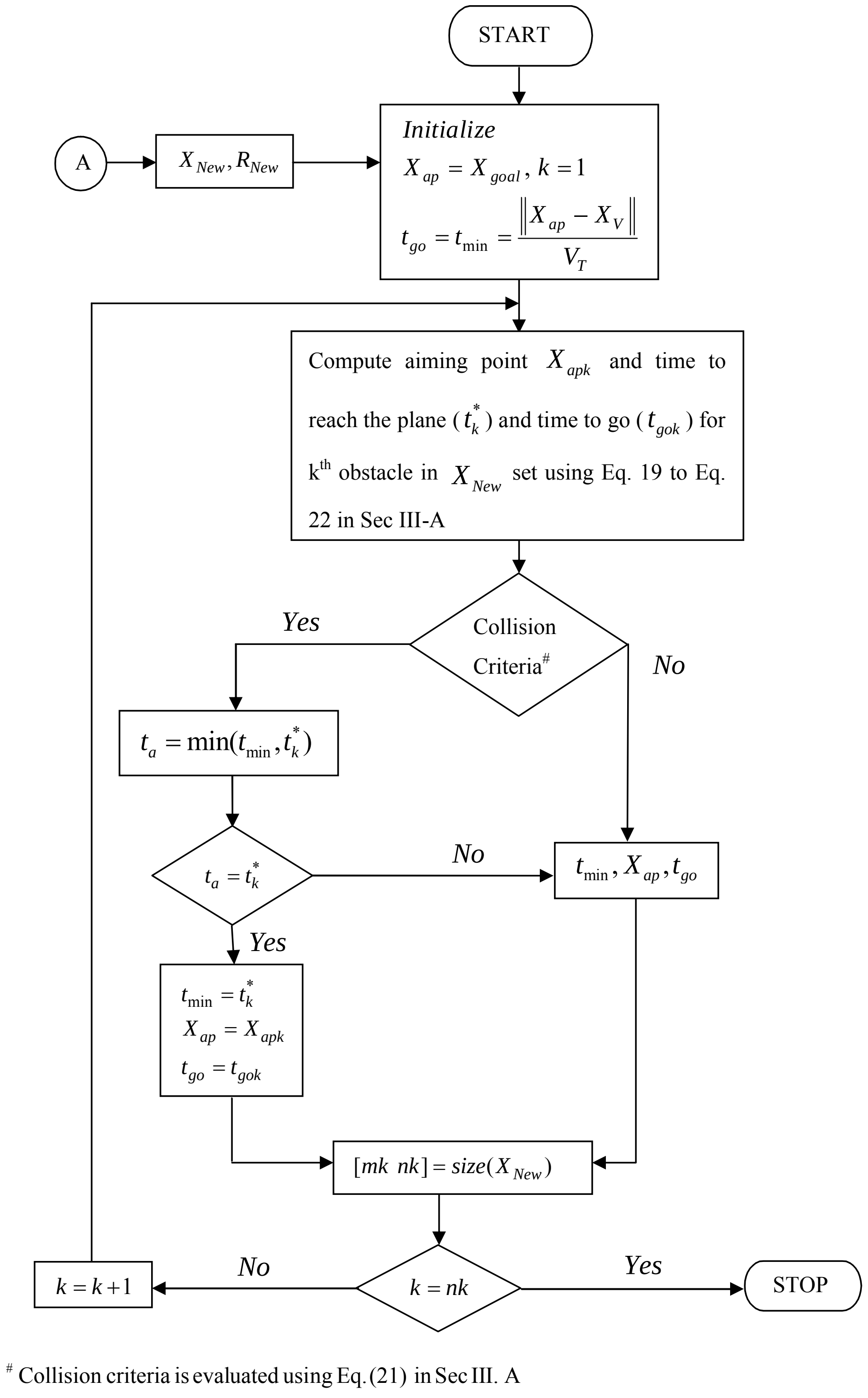

START
A
X_{New}, R_{New}
Initialize
$X_{ap} = X_{goal}, k = 1$
$t_{go} = t_{\min} = \frac{\|X_{ap} - X_V\|}{V_T}$
Compute aiming point X_{apk} and time to reach the plane (t_k^*) and time to go (t_{gok}) for kth obstacle in X_{New} set using Eq. 19 to Eq. 22 in Sec III-A
Collision Criteria#
Yes
No
$t_a = \min(t_{\min}, t_k^*)$
$t_a = t_k^*$
No
Yes
$t_{\min}, X_{ap}, t_{go}$
$t_{\min} = t_k^*$
$X_{ap} = X_{apk}$
$t_{go} = t_{gok}$
$[mk\ nk] = size(X_{New})$
$k = nk$
No
Yes
$k = k + 1$
STOP
# Collision criteria is evaluated using Eq.(21) in Sec III. A

Figure 3.11: Flowchart for Collision Avoidance of Multiple Individual Obstacles.

Chapter 4

Obstacle Position and Size Estimation with Camera

In the previous section, assumption is made that the obstacle is point obstacle and size of the obstacle considering a safety radius around it was assumed spherical. The camera coordinates of the obstacle's projection on cameras image plane are obtained with onboard camera sensors mounted on the UAV and image processors. In this section, the procedure involved in obtaining the pixel data on image plane and extracting the obstacle information is described to make a closer resemblance with realistic approach. The pin hole camera model is considered and assumption made here is that camera is ideal and calibrated. The shape of the obstacle is considered as sphere. The steps followed can be summerised as firstly, Extracting the potential feature points from stereo images using harris detector and then, stereo matching is done along epipolor line. Finally, the 3D information of obstacle obtained by applying triangulation method to the matched feature points.

4.1 Camera Model

In this work, the pinhole camera model is used for describing the geometric relations between 3D object and their projections onto the image plane of the camera and its geometry is shown in Fig 4.1 [46]. The camera axis system, denoted with subscript c, is defined to have its origin at the camera centre C with the X_cY_c plane parallel to the image plane. The intersection point of the Z_c axis in image plane is called the principal point which is close to the centre of

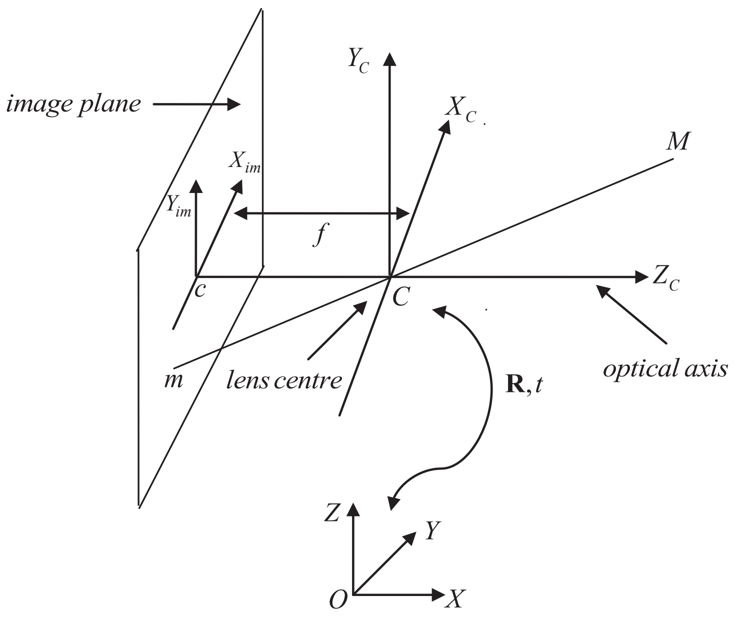


Figure 4.1: The camera coordinate and world coordinate frame

the image plane. The focal length f is defined as the distance between the camera centre and the image plane. The mathematical relation between the camera coordinate to the projected image (*pixel*) coordinate is given by a camera calibration matrix K as follows [46],

$$\mathbf{K} = \begin{bmatrix} s_h & 0 & 0 \\ 0 & s_v & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_h & 0 & p_h \\ 0 & f_v & p_v \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

$$\mathbf{K} = \begin{bmatrix} \alpha_h & 0 & c_h \\ 0 & \alpha_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

where, s_h, s_v are pixel size in the horizontal and vertical direction of the image respectively, K is also called as the intrinsic matrix. α_h and α_v represent the focal length (*in pixels*) in the horizontal and vertical direction respectively. c_h, c_v are the image centre.

let us define the world coordinate system with subscript w. To incorporate the camera position and orientation with respect to world coordinate system, we rotate and translate the point X_c in the camera coordinate system. In Euclidean coordinates this will give us

$$X_c = \mathbf{R}(X_w - C) \tag{4.3}$$

where, $\mathbf{R}$ is a 3×3 rotation matrix and C is the position of the camera centre in Euclidean coordinates. The relation between world coordinate to camera coordinate can be represented in homogeneous coordinates as follows,

$$\begin{bmatrix} x^c \\ y^c \\ z^c \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}C \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} x^w \\ y^w \\ z^w \\ 1 \end{bmatrix} \tag{4.4}$$

The camera matrix (it is also called projection matrix) is given as,

$$P = \mathbf{KR}[\mathbf{I}, -C] \tag{4.5}$$

4.2 Stereo Camera

A stereo vision system is composed of two pinhole cameras, a 3D object and its projection on both image planes shown in Fig. 4.2. Using two or more images of the scene, each acquired from a different viewpoint in space, feature detection and stereo matching of 3D structure is carried out by triangulation [21] method.

4.2.1 Feature detection

The detection of feature points is the determination of interest points in a given image frame. Various methods for finding interest points are available, some of them are the Scale-Invariant Feature Transform (SIFT) [23, 24], the Speeded Up Robust Features (SURF) detector [25], Feature form accelerated segment test (FAST)[26] and the Harris corner [28]. The SIFT feature points are computed by finding the potential feature in different scale space. These features are localized upto the subpixel accuracy and are given in the form of descriptor vector. Each descriptor consists of scale and orientation of the corresponding feature. SIFT features are very accurate and it is invariant to scale, rotation, projective transformation and illumination changes but it's slow to execute and computationally intensive. In SURF detector is computationally fast and less accurate when it's compared to SIFT. It uses hessian matrix which can be calculated using integral images. In this report we have used the Harris corner detector [28], compared to remaining methods it is simple and less computational complexity.

Harris corner detection is used to extract a set of feature points from the left and right images. An interest operator tuned for corner detection is applied to image pair and pixels with the highest interest values selected as features. These potential features are also used for calibration of stereo camera. In this method involves shifting a small patch or window of image in all directions. If the window contains any corner point then shifting the window along all direction results in large change in its intensity values.

$$E(u,v) = \sum_{x,y} W_{x,y} |I_{u+x,v+y} - I_{x,y}|^2 \tag{4.6}$$

$I_{x,y}$ intensity value at point (x, y), u, v introduce a small shift amount of the window. The window is chosen as a gaussian window $W_{x,y}$. Using the Taylor series, the expression

$I_{u+x,v+y}$ in above equation can be rewritten as,

$$E(u,v) = \sum_{x,y} W_{x,y} |I_{x,y} + uI_x + vI_y - I_{x,y}|^2 \tag{4.7}$$

$$E(u,v) = [u,v] M \begin{bmatrix} u \\ v \end{bmatrix} \tag{4.8}$$

,

where M is called covariance matrix,

$$M = \sum_{x,y} W_{x,y} \begin{bmatrix} I_{x,x} & I_{x,y} \\ I_{x,y} & I_{y,y} \end{bmatrix} \tag{4.9}$$

The quality of the corner can be measured by the following response function,

$$R = Det(M) - K(Trace(M))^2 \tag{4.10}$$

The response function R only depends on the eigen values of M. For corner points it will produce the large positive value ($R > 0$), for edge region it will give the large negative value ($R < 0$) and for flat region it will give small $|R|$ value.

4.2.2 Stereo matching

Stereo matching algorithm is used to find correspondences between the extracted feature points between left and right images which gives relative position and orientation between stereo cameras (i.e. Fundamental matrix).

4.2.3 Epipolar constraint

In this section, the epipolar geometry [21] is presented. Consider two image planes with the distinct viewpoints as in Fig. 4.2. Let x_l, y_l and x_r, y_r be the corresponding feature points in two image planes. The epipolar geometry defines the geometric relationship between these corresponding points. The plane passing through the camera center C_l and C_r and the 3D world point x^w is called an epipolar plane. The projection $e(e')$ of one camera center onto the image plane of the other camera frame is called an epipole. An epipolar line $l(l')$ is the

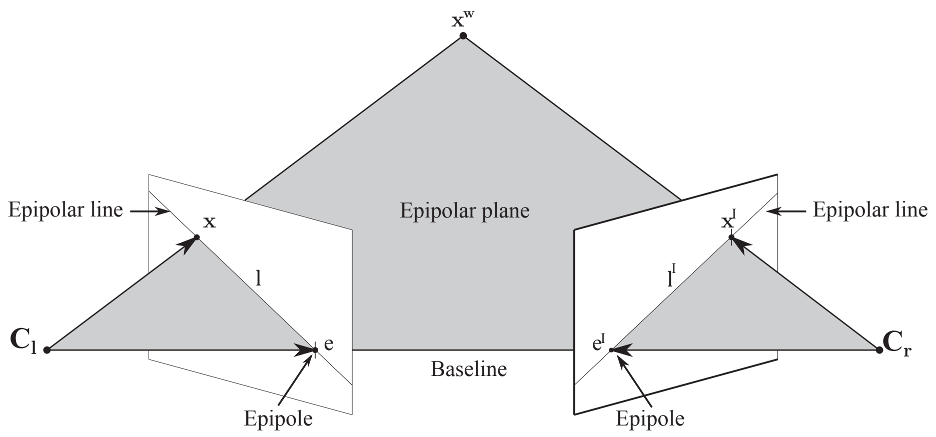


Figure 4.2: The geometry of stereo camera

intersection of an epipolar plane for x^w with the image plane. All epipolar lines pass through the epipole.

4.2.4 Fundamental Matrix

The fundamental matrix represents the epipolar geometry algebraically and contains most of the information about the relative position and orientation between the two views. If F is the fundamental matrix [42, 44], then F^T is the matrix which satisfies

$$X'^T F X = 0 \tag{4.11}$$

for all corresponding points x and x' in both left and right image planes. For any point x in the first image, the corresponding epipolar line is $l' = Fx$. Similarly, $l = F^T x'$ represents the epipolar line corresponding to x' in the second image plane. For any point x(other than epipole e), the epipolar line $l' = Fx$ contains the epipole e'. Thus e' satisfies $e'^T Fx = 0$ for all x.

In this work, we have used SSD-based matching by searching both the images to minimize the similarity criterion. The similarity criteria can be represented by,

$$SSD(x, y) = \sum_{(i,j)\epsilon W} (I_{left}(i, j) - I_{right}(x + i, y + j))^2 \tag{4.12}$$

The stereo matching is done strictly along the epipolar line with only a few pixels to offset above and below it. RANSAC [22],[27] is used to reject outliers.

4.3 3D Reconstruction

The relationship between a point in the scene and its corresponding point in the camera and the projector image coordinates can be written for left and right images as follows [47],

$$\begin{bmatrix} w^1 x^{p^1} \\ w^1 y^{p^1} \\ w^1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x^w \\ y^w \\ z^w \\ 1 \end{bmatrix} \tag{4.13}$$

$$\begin{bmatrix} w^2 x^{p^2} \\ w^2 y^{p^2} \\ w^2 \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \end{bmatrix} \begin{bmatrix} x^w \\ y^w \\ z^w \\ 1 \end{bmatrix} \tag{4.14}$$

$x^{p^1} \cong Px^w, \ x^{p^2} \cong Qx^w$

Where, P and Q denote the projection matrix of the camera and the projector respectively, and the superscripts p^1 *and* p^2 mean the pixel coordinate frames of the camera and w^1 *and* w^2 denotes world frame projector respectively. If a pair of corresponding points in two images can be found, it suggests that they are the projection of a common 3D point. This can be Reconstruction by Singular Value Decomposition (SVD) or inverse pseudo spectral method. In our work SVD is used for finding object points. Finally it (Eqn 4.13 and 4.14) is expressed in terms of homogeneous coordinates.

$$Mx^w = 0$$

$Where\ M =$

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \tag{4.15}$$

$$
\begin{aligned}
m_{11} &= p_{11} - x^{p^1} p_{31} \ , \ m_{12} = p_{12} - x^{p^1} p_{32} \\
m_{13} &= p_{13} - x^{p^1} p_{33} \ , \ m_{14} = p_{14} - x^{p^1} p_{34} \\
m_{21} &= p_{21} - y^{p^1} p_{31} \ , \ m_{22} = p_{22} - y^{p^1} p_{32} \\
m_{23} &= p_{23} - y^{p^1} p_{33} \ , \ m_{24} = p_{24} - y^{p^1} p_{34} \\
m_{31} &= q_{11} - x^{p^2} q_{31} \ , \ m_{32} = q_{12} - x^{p^2} q_{32} \\
m_{33} &= q_{13} - x^{p^2} q_{33} \ , \ m_{33} = q_{14} - x^{p^2} q_{34} \\
m_{41} &= q_{21} - y^{p^2} q_{31} \ , \ m_{42} = q_{22} - y^{p^2} q_{32} \\
m_{43} &= q_{23} - y^{p^2} q_{33} \ , \ m_{44} = q_{24} - y^{p^2} q_{34}
\end{aligned}
$$

Thus, we can estimate the point through singular value decomposition (SVD) related techniques. The four elements of the last column of V obtained by SVD of M (UDV^T) are the homogeneous coordinates of x^w.

4.4 Simulation Result

In this section, point mass model of UAV is considered. Simulations are carried out in Matlab's VRML environment. For simplicity, obstacle shape with safety radius is taken as spherical. Virtual stereo camera is mounted on UAV in VRML environment. The base length B between two cameras is taken as $30cm$. Field of view of horizontal and vertical direction is taken as $fov = 60°$. Resolution of the virtual camera is 576×380 and the focal length (in pixel) can be computed using following expression.

$$\alpha_h = f_h s_h = \frac{0.5 * width}{tan(fov * 0.5)} * \frac{576}{width} \tag{4.16}$$

$$\alpha_v = f_v s_v = \frac{0.5 * height}{tan(fov * 0.5)} * \frac{380}{height} \tag{4.17}$$

The camera center is taken as c_h=288, c_v=190.

4.4.1 Single Obstacle

The simulation is carried out with stationary single obstacle and the initial position of UAV is $[20, 10, 10]$, Initial Velocity is $[10, 1.2, 1.0]$, Target Position is $[150, 22, 22]$. The feature points detection algorithm results are shown in Fig. 4.3. The stereo matching is done by searching the minimum sum of square difference (SSD) of potential features along the epipolor line and outliers are removed by RANSAC algorithm shown in Figs. 4.4. The 3D-Reconstruction of the potential features are shown in Fig. 4.5. The approximate value of the obstacle centre can be obtained by finding the mean of the reconstructed 3D-potential feature points. However obstacle centre may not be an exact one but due to 3D symmetrical nature of the sphere the calculation of centre is closed to the actual one. Radius of the sphere can be computed by taking average distance between computed centre and all feature points. Moreover, if the UAV is approaching closer to the obstacle it may get only partial information about the obstacle because of the restriction of its field of view . To avoid this problem the value of the obstacle centre and radius is freezed from some threshold distance between UAV and the computed obstacle. In this simulation we got obstacle centre as $[110.46, 14.52, 6.62]$ and radius is $15.15m$.

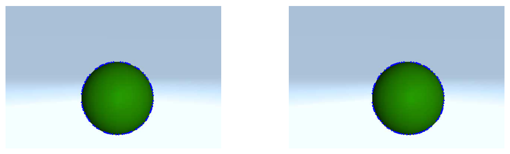


Figure 4.3: Feature points detected by Harris corner

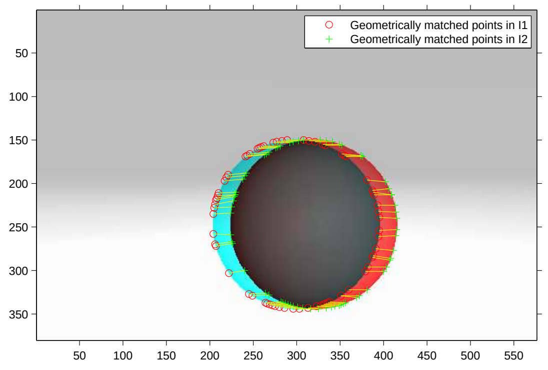


Figure 4.4: SSD based stereo matching

The 3D trajectory of UAV is shown in Fig. 4.6. The guidance command such as Angle of attack and Bank angle profile is shown in Figs 4.7 and 4.8 respectively. The Thrust profile is shown in Fig. 4.9

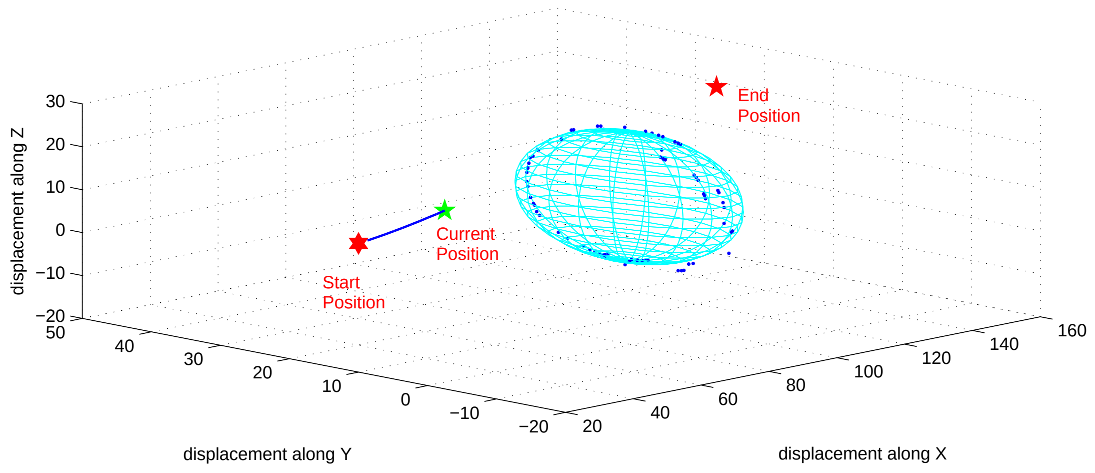


Figure 4.5: 3D reconstruction of matched feature points

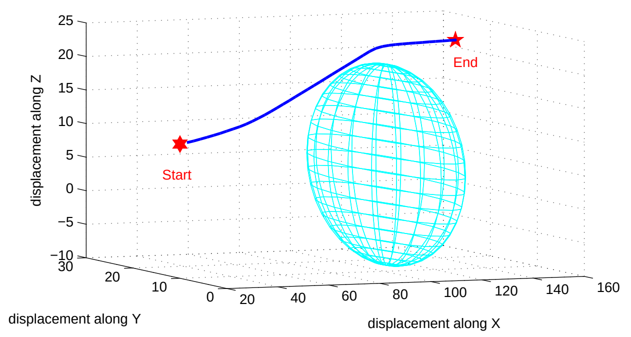


Figure 4.6: UAV Trajectory

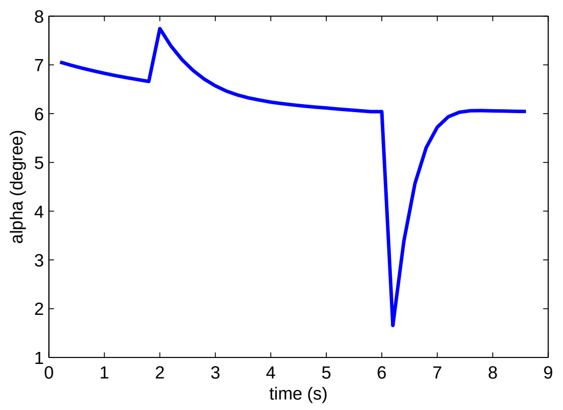


Figure 4.7: Angle of attack Profile

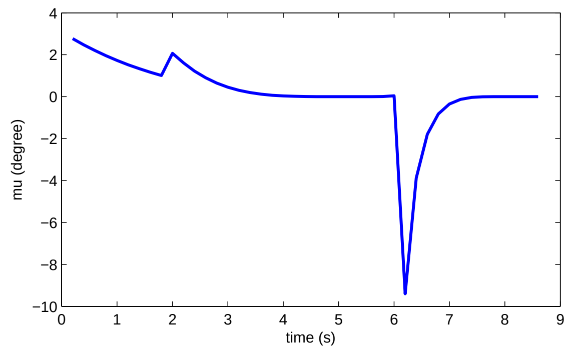


Figure 4.8: Bank angle Profile

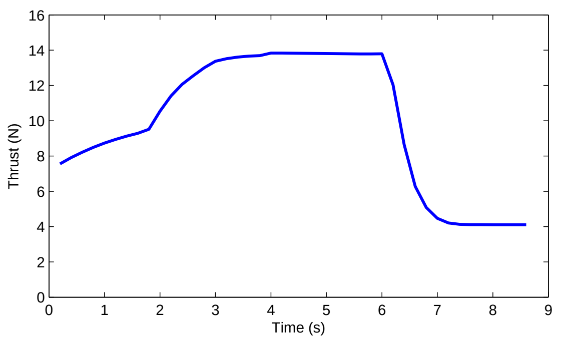


Figure 4.9: Thrust Profile

Chapter 5

Simulation Results

5.1 Simulation Results

Simulations are carried out to test the performance of reactive collision avoidance algorithm. Both the Kinematic model and Point mass model of UAV is considered for the system dynamics. The aerodynamic data for the model is taken from the AE-2 model of the UAV as shown in Fig.A.1. Single stationary obstacle, single moving obstacle with constant velocity and multiple stationary obstacles are considered for simulation study. Validation of the reactive collision avoidance algorithm is also carried out with randomized simulations. For extracting the accurate data from noisy sensor measurements EKF as well as UKF estimation techniques are used which estimate the obstacle position as well as obstacle velocity. In order to have similarity with practical stereovision sensors the separation between the two sensors is 0.40 meter and focal length is 0.22 centimeter is considered for both the sensors.

5.2 Kinematic model of UAV with EKF Estimation

In this section Kinematic model of UAV is considered and camera sensor model is assumed to be noisy and the camera sensor state estimation is carried out using Extended Kalman Filter. The simulation is carried out with single stationary as well as single moving obstacle with constant velocity. The objective here is to compute the estimated position of the obstacle and use the guidance command to avoid the collision with in the available time to go and guide the UAV to appropriate destination.

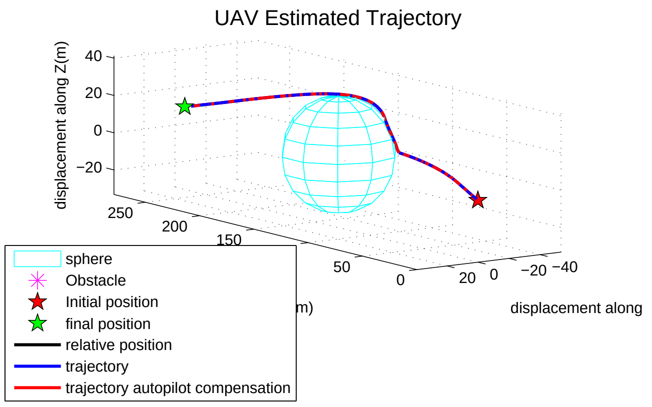


Figure 5.1: UAV Kinematic model with EKF Single Stationary Obstacle: UAV Trajectory

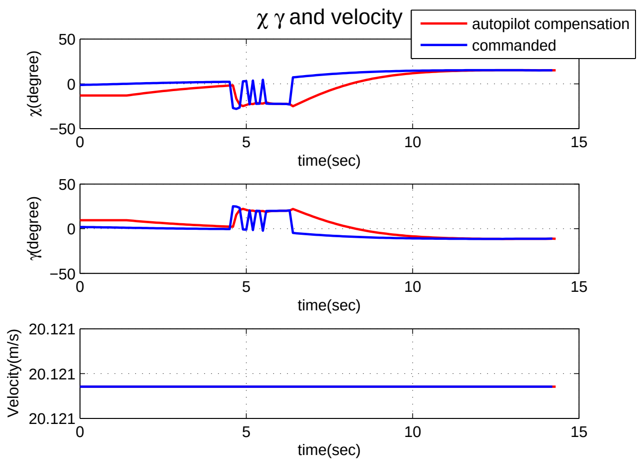


Figure 5.2: UAV Kinematic model with EKF Single Stationary Obstacle: χ, γ and Velocity

5.2.1 Single obstacle

Case 1: Stationary Obstacle

The simulation is carried out with single stationary obstacle the initial position of UAV is $[0, 0, 0]$, Initial Velocity $[19.33, -4.5, 3.3]$, Target Position $[279, -6.3, 9.2]$, Obstacle position $[138, -7.0, 4.0]$ and UAV final location $[274, -7.4, 10.0]$. Kinematic model of UAV is considered and Extended Kalman Filter is used for sensor state estimation. Fig. 5.1 shows the trajectory of the UAV. The guidance commands for the UAV with kinematic model is shown in Fig. 5.2. Autopilot lag compensation is provided for the guidance command and corresponding trajectory is also plotted which almost overlaps the actual trajectory. Velocity profile of the UAV is constant throughout the simulation. The guidance commands χ and γ are tracking the desired guidance command.

The sigma error bounds for stereovision camera sensor states r, χ and γ are given as Fig. 5.3. The plot shows that errors are within the bounds. The UAV to target distance and the UAV to obstacle distance is shown in Fig. 5.4 where the UAV to obstacle distance is firstly decreasing and further increasing as the UAV first approaches towards the obstacle and then avoids it by executing the guidance cammand and moves towards the destination. However, the distance to target is constantly decreasing. Simulation stopping criteria is applied when the UAV reaches within $5m$ range of the destination.

Case 2: Moving Obstacle

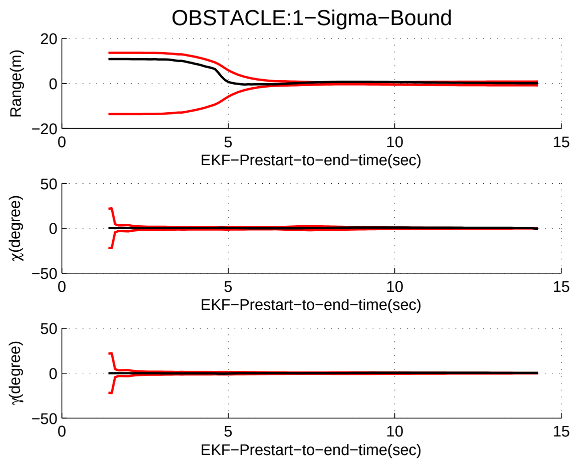


Figure 5.3: UAV Kinematic model with EKF Single Stationary Obstacle: Sigma error bound

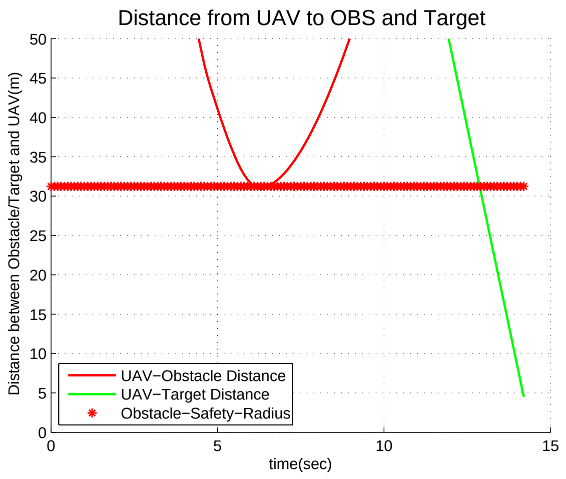


Figure 5.4: UAV Kinematic model with EKF Single Stationary Obstacle: UAV to Obstacle and Target Distance

The simulation is carried out with single obstacle moving with constant velocity $[2, 0, 0]$, the Initial position of UAV is $[0, 0, 0]$, UAV Initial Velocity is $[19.9, 0.97, 0.99]$, Target Position is $[389.9, -3.0, -3.0]$, Obstacle initial position is $[120.0, -0.54, -0.51]$, Obstacle initial position is $[159.10, -0.54, -0.51]$ and UAV final location is $[386.9, -2.6, -2.7]$. Kinematic model of UAV is considered and extended kalman filter is used for estimation. The objective here is to compute the estimated position as well as the estimated velocity of the obstacle and use the guidance command for the reactive collision avoidance with the obstacle in the available time to go. In the moving scenario the obstacle is considered non-cooperative and the concept of miss distance [7] is used. Fig. 5.5 shows the trajectory of the UAV. The trajectory with autopilot compensation is also plotted. The UAV is able to avoid collision with moving obstacle.

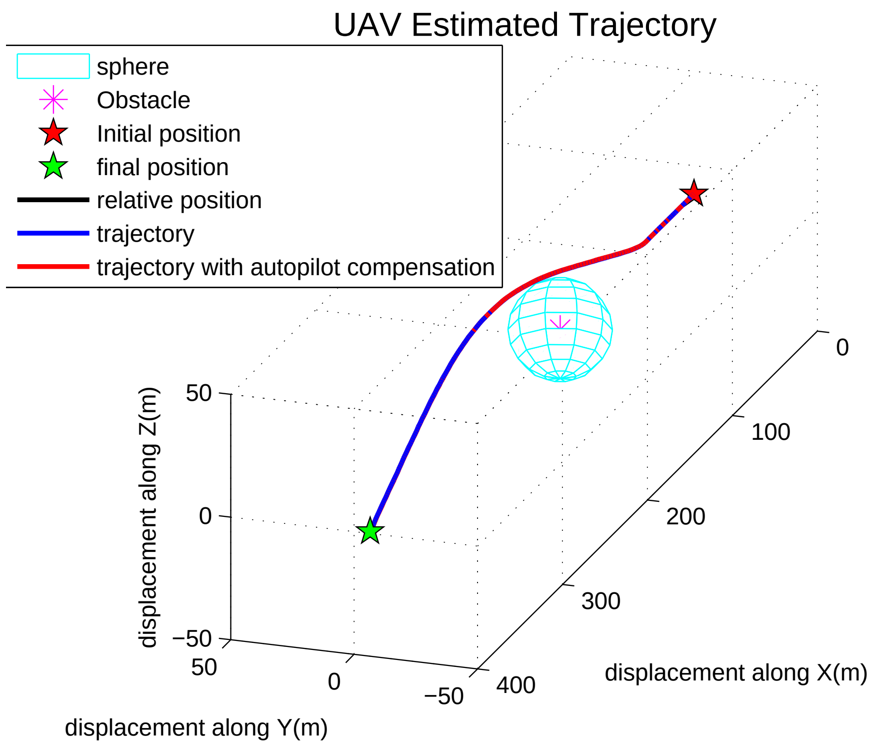


Figure 5.5: UAV Kinematic model with EKF Single Moving Obstacle: UAV Trajectory

The guidance commands for the UAV with kinematic model is shown in Fig. 5.7. The guidance command is tracking the desired command. The following Fig. 5.6 shows autopilot compensation provided for guidance command. Sigma error bounds for stereovision camera sensor states are given as Fig. 5.8 which are well with in bounds. The UAV to target distance and the UAV to obstacle distance profile is shown in Fig. 5.9

The obstacle is assumed to be moving with constant velocity and based on successive data obtained from stereovision camera sensors in their coordinate frames, using camera sensor state dynamics and using the Extended Kalman Filter estimation, the velocity of the obstacle is estimated and as shown in the Fig.5.10. The velocity estimate becomes erroneous as the camera sensor comes to closer to obstacle. This can be correlated with the time instants shown in the fig.5.11 where the forward direction(x-direction) position components of UAV and obstacle are plotted.

It is observed that the guidance command is efficiently able to perform the reactive collision avoidance in single obstacle scenario when the obstacle is stationary as well as moving with constant velocity.

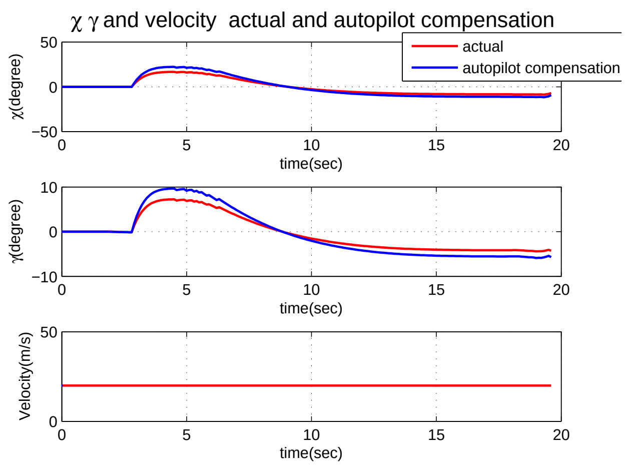


Figure 5.6: UAV Kinematic model with EKF Single Moving Obstacle: Autopilot Compensation

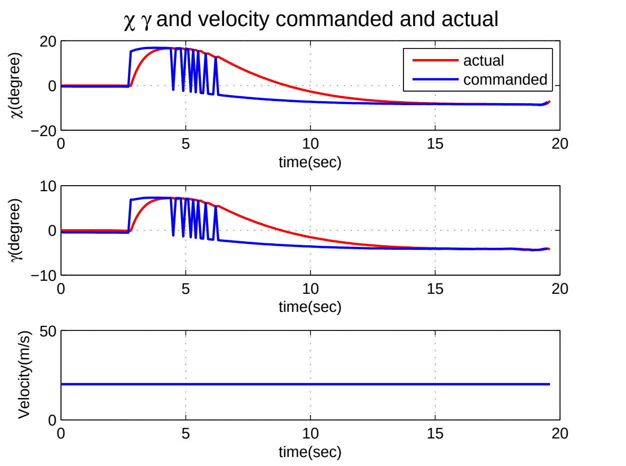


Figure 5.7: UAV Kinematic model with EKF Single Moving Obstacle: χ, γ and Velocity

5.3 Kinematic model of UAV with UKF Estimation

In this section Kinematic model of UAV is considered and camera sensor model is assumed noisy and the camera sensor state estimation is carried out using Unscented Kalman Filter(UKF). The UKF's excellence over EKF for nonlinear systems in noisy conditions are exploited and the simulation results are presented for single stationary obstacle environment.

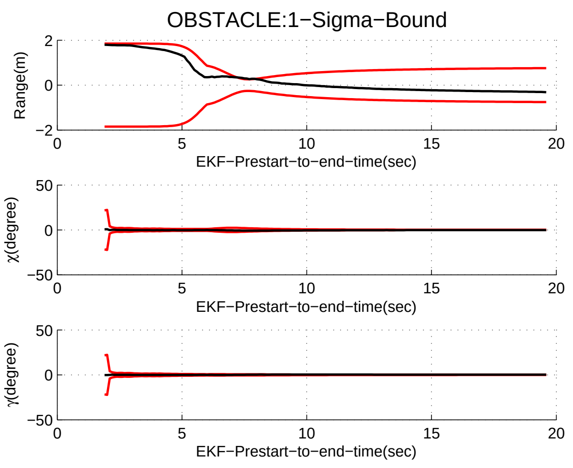


Figure 5.8: UAV Kinematic model with EKF Single Moving Obstacle: Sigma error bound

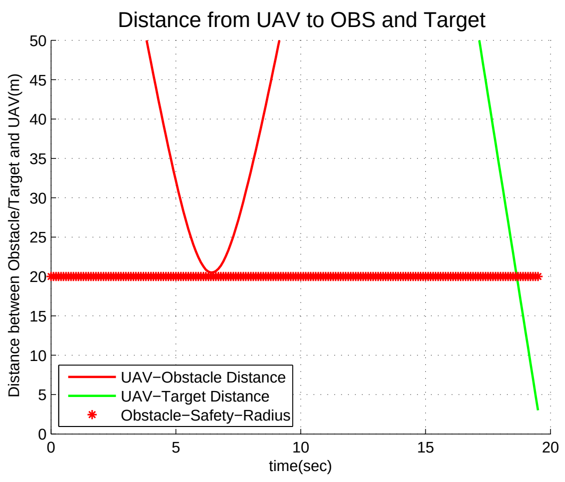


Figure 5.9: UAV Kinematic model with EKF Single Moving Obstacle: UAV to Obstacle and Target Distance

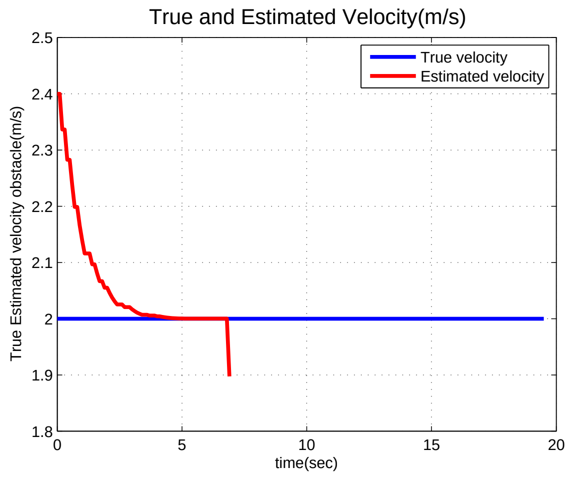


Figure 5.10: UAV Kinematic model with EKF Single Moving Obstacle: True and Estimated Velocity of Obstacle

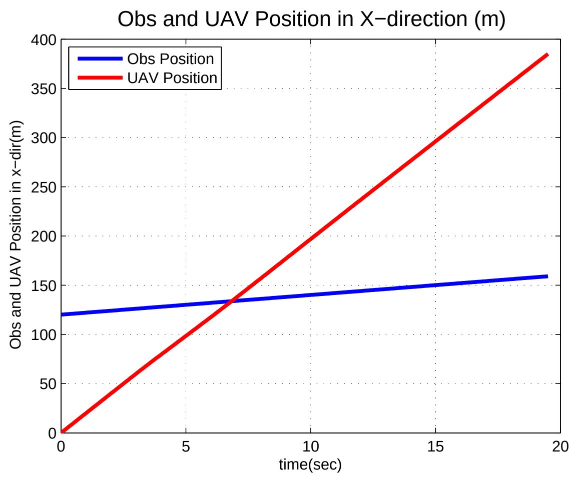
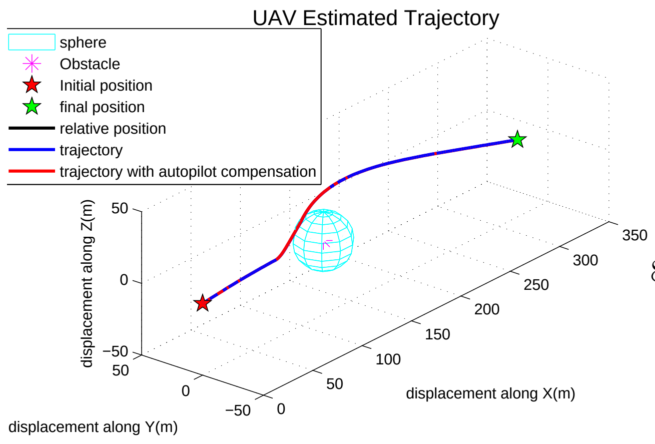
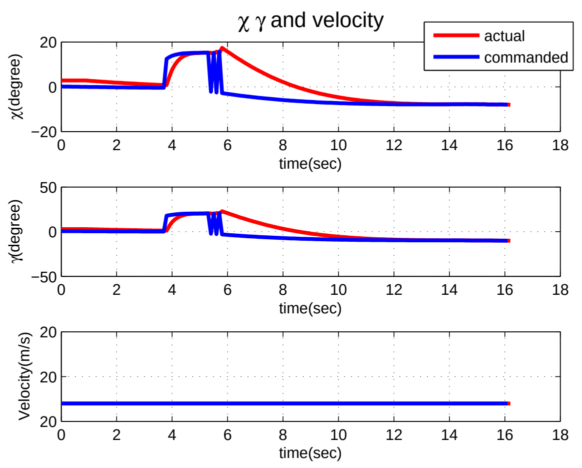


Figure 5.11: UAV Kinematic model with EKF Single Moving Obstacle: Obstacle and UAV's X-direction Position



8.5cm



8.5cm

Figure 5.12: UAV Kinematic model with UKF Single Stationary Obstacle: UAV Trajectory

Figure 5.13: UAV Kinematic model with UKF Single Stationary Obstacle: χ, γ and Velocity

5.3.1 Single obstacle

The simulation is carried out with single stationary obstacle the initial position of UAV is $[0, 0, 0]$, Initial Velocity $[19.95, 0.97, 0.99]$, Target Position $[319.9, 0.65, 3.3]$, Obstacle position $[123.85, 1.21, 0.78]$ and UAV final location $[316.35, 1.16, 4.0]$ kinematic model of UAV is considered and Unscented kalman filter is used for estimation. Fig. 5.12 shows the trajectory of the UAV with actual guidance as well as guidance with autopilot compensation. The guidance commands for the UAV with kinematic model is shown in Fig.5.13 where tracking of the desired guidance command is shown. The velocity of the UAV is considered to be constant throughout the simulation.

The autopilot compensation for lag in guidance command is provided in Fig.5.14. The UAV to target distance and the UAV to obstacle distance is shown in Fig. 5.15. It is observed that the guidance command is efficiently able to perform the collision avoidance in single stationary obstacle scenario with kinematic model of UAV and with UKF estimation.

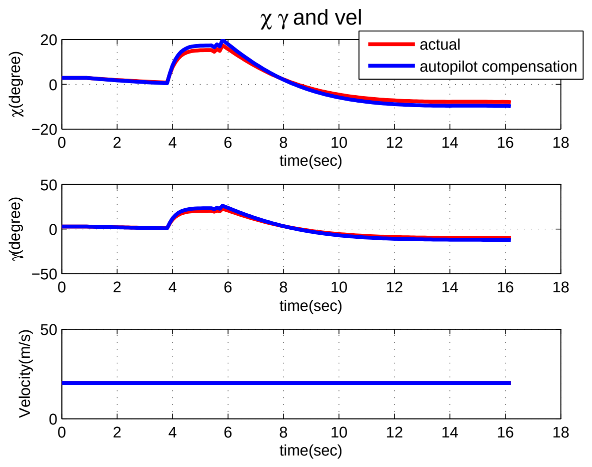
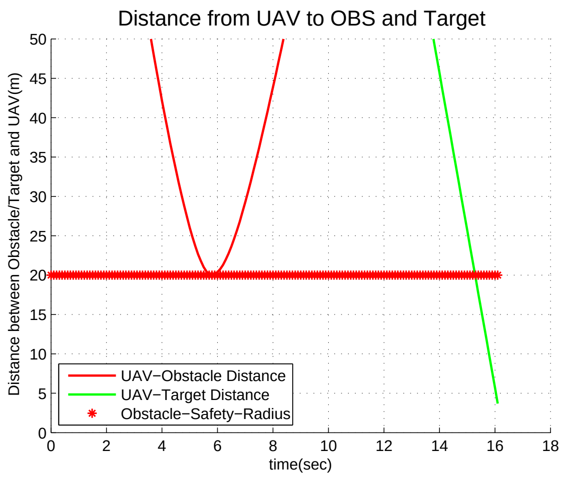


Figure 5.14: UAV Kinematic model with UKF Single Stationary Obstacle: Autopilot Compensation



8.5cm

Figure 5.15: UAV Kinematic model with UKF Single Stationary Obstacle: UAV to Obstacle and Target Distance

5.4 Point mass model of UAV with EKF Estimation

In this section Point mass model of UAV is considered and stereovision camera sensor model is assumed noisy and the camera sensor state estimation is carried out using Extended Kalman Filter. The simulation is carried out with single stationary obstacle, single moving obstacle with constant velocity as well as multiple stationary obstacle. The objective here is to compute the estimated position of the obstacle as well as estimated velocity of the obstacle (for moving obstacle only) and use the guidance command to avoid the collision in the available time to go.

5.4.1 Single obstacle

Case 1: Stationary Obstacle

The simulation is carried out with single stationary obstacle the initial position of UAV is $[0, 0, 50]$, Initial Velocity $[19.95, 0.97, 0.99]$, Target Position $[267.0, -4.0, 46.0]$, Obstacle position $[169.95, -1.50, 48.40]$ and UAV Final location $[266.6, -2.4, 46.0]$. Point mass model of UAV is considered and Extended Kalman Filter is used for estimation. Fig. 5.16 shows the trajectory of the UAV obtained with actual guidance command as well as guidance with autopilot compensation. The guidance commands for the UAV with point mass model is shown in Fig. 5.17 where the desired guidance command is computed through the aiming point and actual guidance tracks the desired guidance command. Keeping velocity constant the angle commands are used to achieve the aiming point.

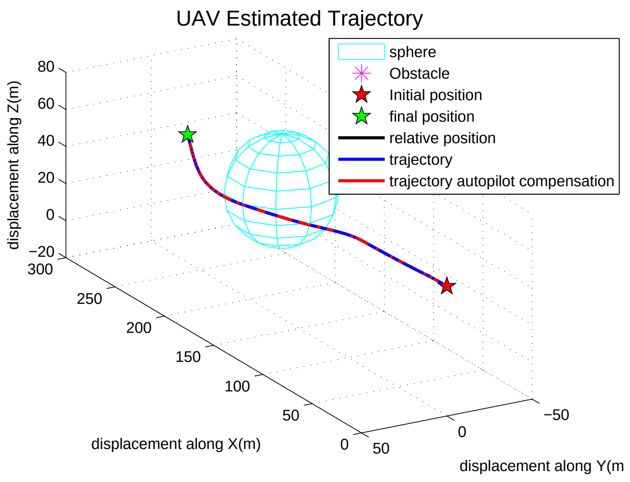


Figure 5.16: UAV Point mass model with EKF Single Stationary Obstacle: UAV Trajectory

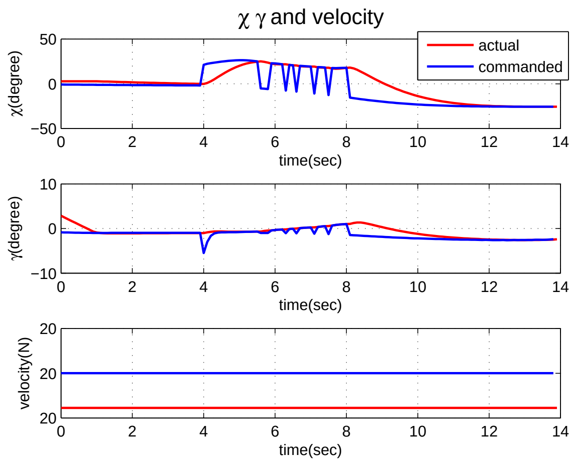


Figure 5.17: UAV Point mass model with EKF Single Stationary Obstacle: χ, γ Velocity

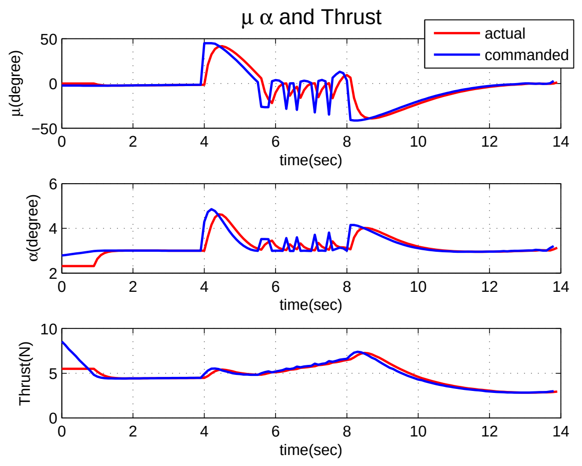


Figure 5.18: UAV Point mass model with EKF Single Stationary Obstacle: μ, α and Thrust

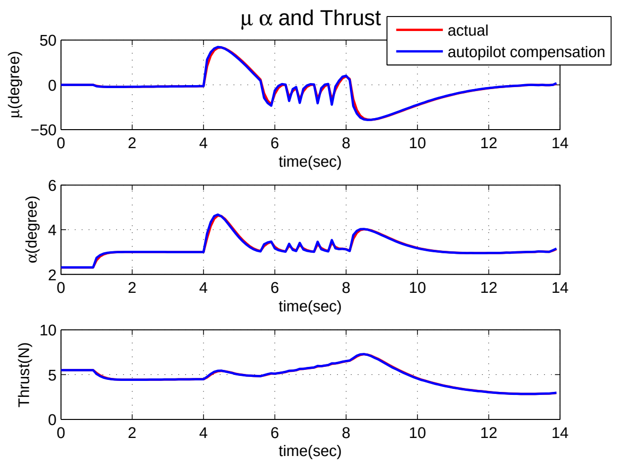


Figure 5.19: UAV Point mass model with EKF Single Stationary Obstacle: Autopilot Compensation

The physical guidance commands α, μ and thrust which are in turn computed using the guidance commands χ, γ and velocity is shown in Fig. 5.18. Since the UAV model is an aerodynamic model the physical guidance commands are more relevant. The actual guidance command closely tracks the commanded value. The autopilot compensation for lag in guidance command is shown in Fig. 5.19. The UAV to target distance and the UAV to obstacle distance is shown in Fig. 5.21. The sigma error bounds for stereovision camera sensor states are given as Fig.5.20.

The simulation results show that UAV with point mass model is able to perform the reactive collision avoidance with the stationary obstacle.

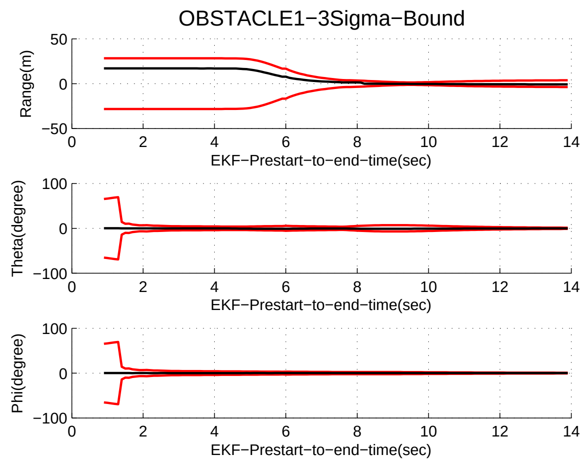


Figure 5.20: UAV Point mass model with EKF Single Stationary Obstacle: Sigma error bound

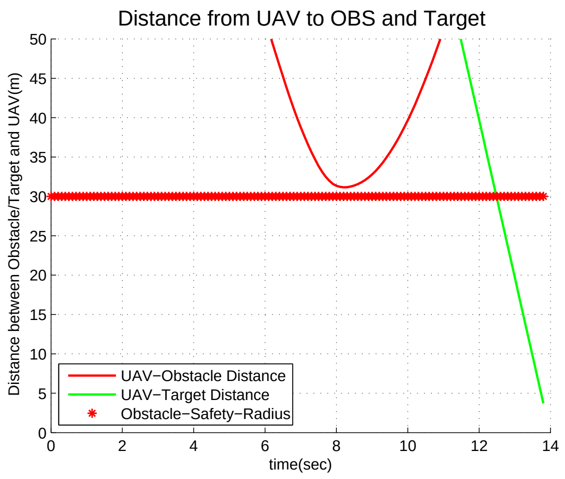


Figure 5.21: UAV Point mass model with EKF Single Stationary Obstacle: UAV to Obstacle and Target Distance

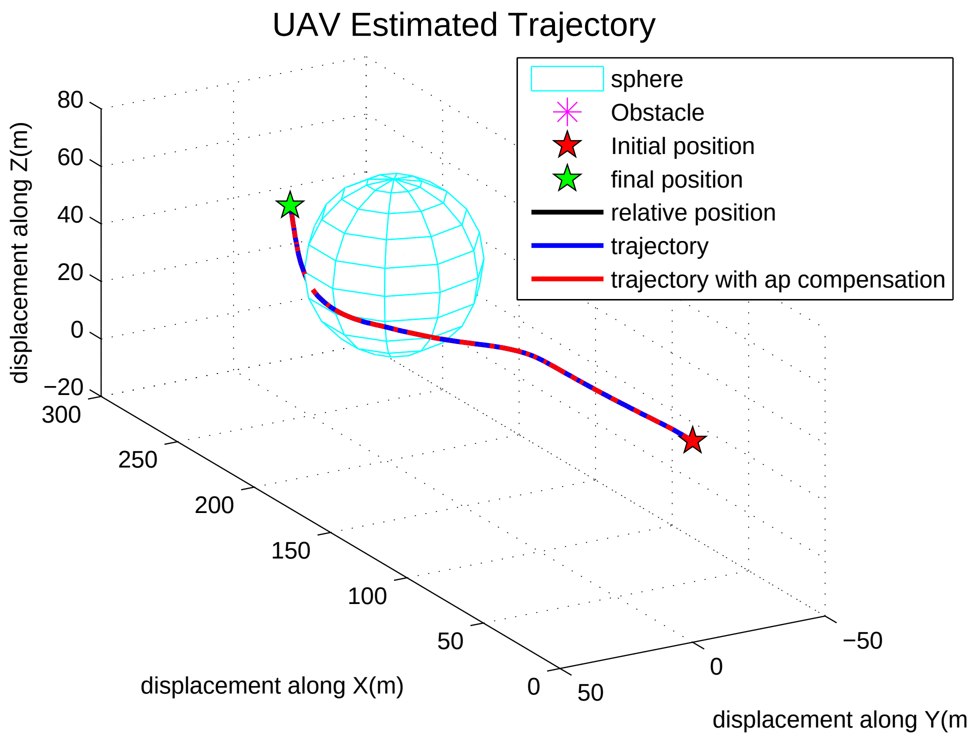


Figure 5.22: UAV Point mass model with EKF Single Moving Obstacle: UAV Trajectory

Case 2: Moving Obstacle

The simulation is carried out with single obstacle moving with constant velocity. Obstacle velocity $[2, 0, 2]$, the Initial position of UAV is $[0, 0, 50]$, Initial Velocity $[19.95, 0.97, 0.99]$, Target Position $[269.95, -4.0, 46.0]$, Obstacle initial position $[169.95, -1.52, 48.4]$, Obstacle final position $[197.7, -1.52, 48.49]$ and UAV Final location $[266.80, -2.50, 46.0]$. Point mass model of UAV is considered and extended kalman filter is used for stereovision camera state estimation. Fig. 5.22 shows the trajectory of the UAV obtained with guidance command as well as guidance with autopilot compensation. The guidance commands for the UAV with point mass model is shown in Fig. 5.23 where the actual guidance command is tracking the desired command. Since it is being a moving obstacle scenario and the trajectory plot shows only the final location of the obstacle, the perturbation in the commanded guidance is due to the UAV's close proximity with the obstacle safety sphere.

The physical guidance command is shown in Fig. 5.24 which is showing the α, μ and thrust profiles for averting the collision with a moving obstacle. The autopilot compensation for the physical guidance command is shown in Fig. 5.25. The UAV to target distance and

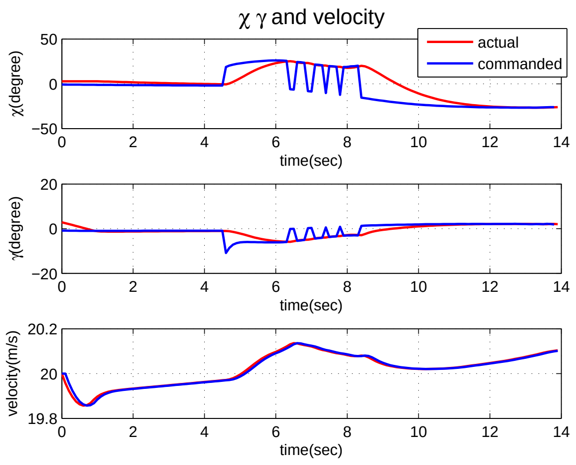


Figure 5.23: UAV Point mass model with EKF Single Moving Obstacle: χ, γ Velocity

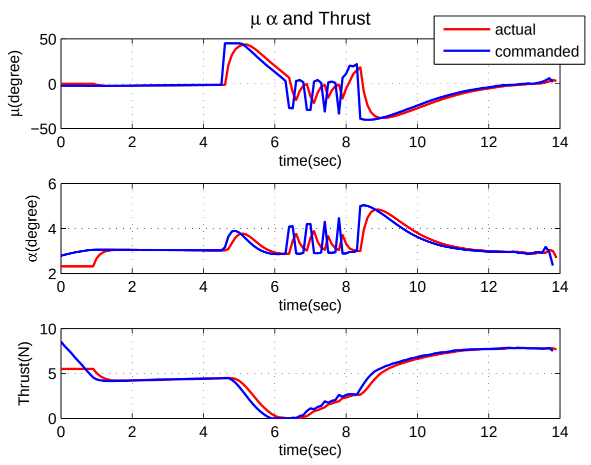


Figure 5.24: UAV Point mass model with EKF Single Moving Obstacle: μ, α and Thrust

the UAV to obstacle distance profiles are shown in Fig. 5.26.

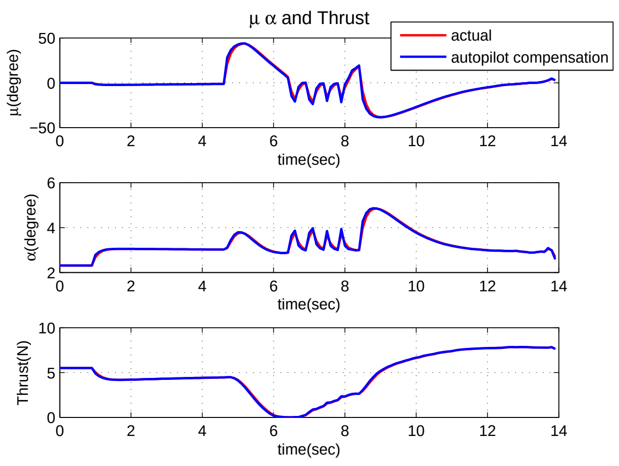


Figure 5.25: UAV Point mass model with EKF Single Moving Obstacle: Autopilot Compensation

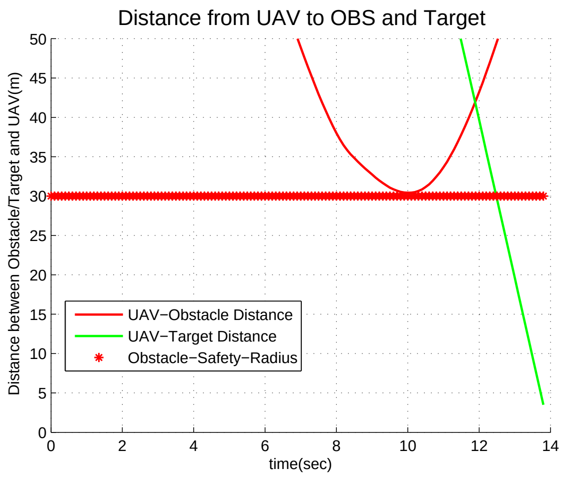


Figure 5.26: UAV Point mass model with EKF Single Moving Obstacle: UAV to Obstacle and Target Distance

The obstacle is assumed to be moving with constant velocity and based on successive data obtained from stereovision camera sensors in respective camera coordinate frames, using sensor state dynamics and with the extended kalman filter estimation, the velocity of the obstacle is estimated and as shown in the Fig.5.27. The velocity estimate becomes erroneous as the camera sensor comes to closer to obstacle. This can be correlated with the time instants shown with Fig.5.28 where the forward direction(x-direction) position components of UAV and obstacle are plotted.

The simulation results show that UAV with point mass model dynamics is able to perform reactive collision avoidance in moving obstacle scenario.

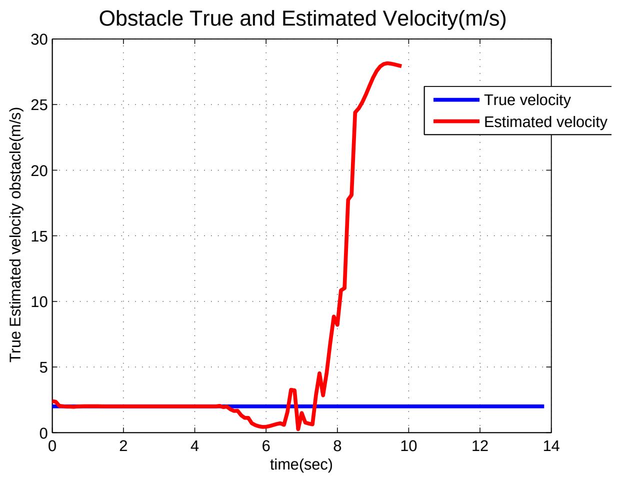


Figure 5.27: UAV Point mass model with EKF Single Moving Obstacle: Obstacle True and Estimated Velocity

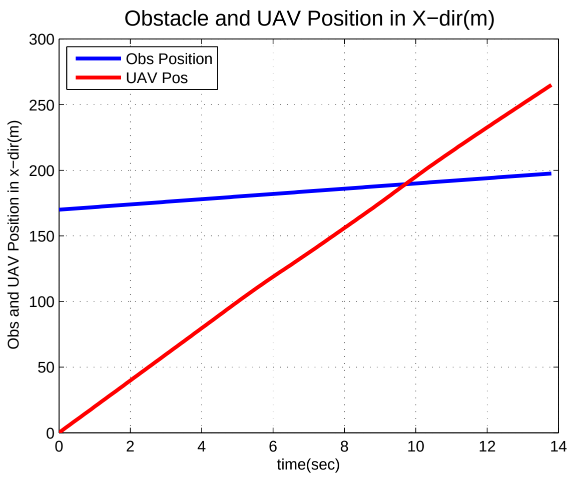


Figure 5.28: UAV Point mass model with EKF Single Moving Obstacle: Obstacle and UAV Position in x-direction

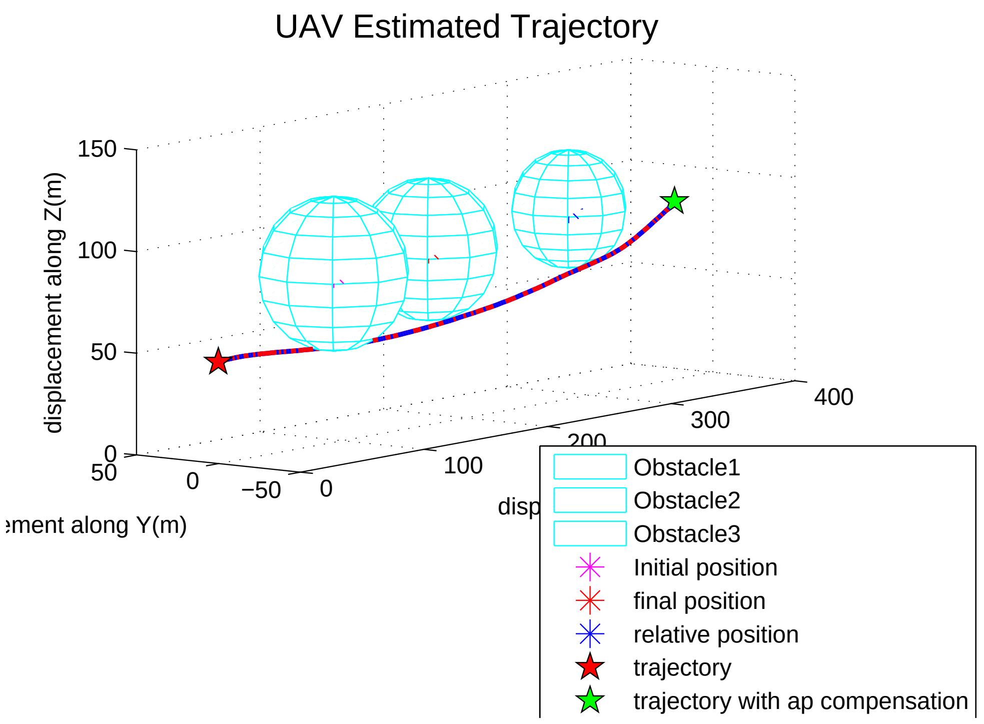


Figure 5.29: UAV Point mass model with EKF Multi Stationary Obstacle: UAV Trajectory

5.4.2 Multiple obstacle

In order to test the performance of the reactive collision avoidance algorithm in multiple stationary obstacle scenario, the simulation is carried out with three stationary obstacles with Initial positions [89.97,-2.5,83.49], [179.97,7.48,84.49] and [299.97,12.48,90.49]. The initial position of UAV is [0,0,50], Initial Velocity of UAV [19.95,0.97,0.99], Target Position [362.97,-4.51,88.49], UAV final position [359.57,-4.44,86.40]. Point mass model of UAV is considered with EKF Estimation. Fig. 5.29 shows the trajectory of the UAV obtained through actual guidance command as well as guidance with autopilot compensation. In order to provide a clear view of UAV's closest approach with the three obstacles, the UAV to target distance and the UAV to three different obstacles distance is shown in Fig. 5.30. The guidance commands for the UAV with point mass model is shown in Fig. 5.31 with actual and commanded guidance profiles for χ,γ and velocity.

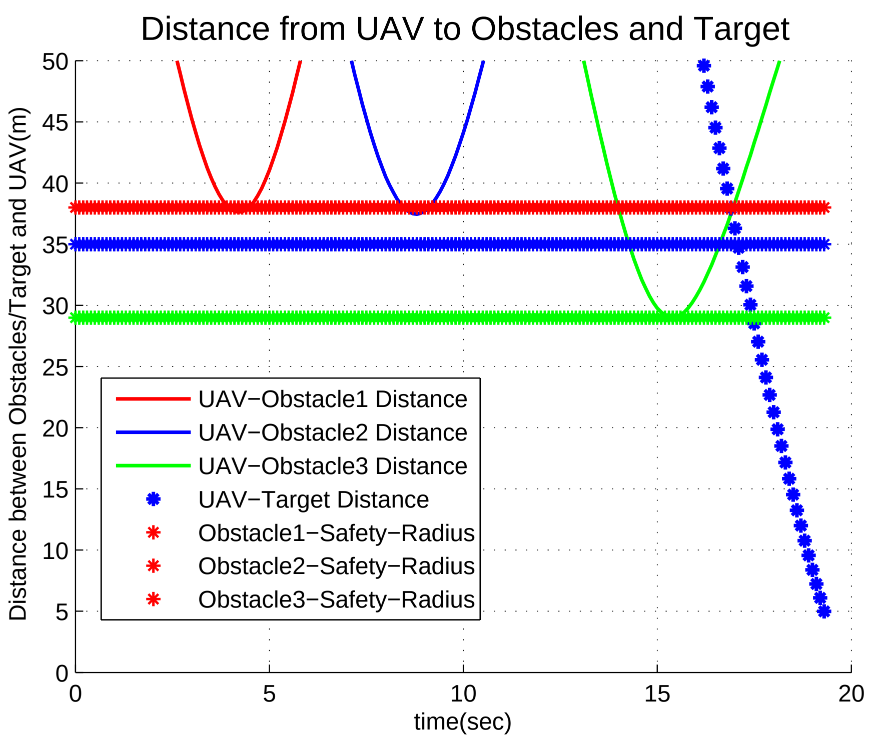
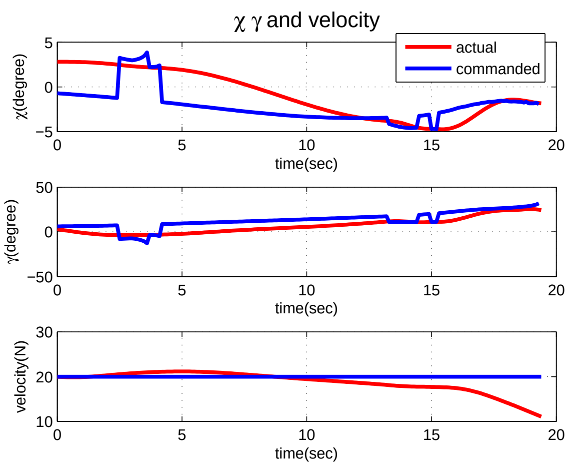


Figure 5.30: UAV Point mass model with EKF Multi Stationary Obstacle: UAV to Obstacles and Target Distance



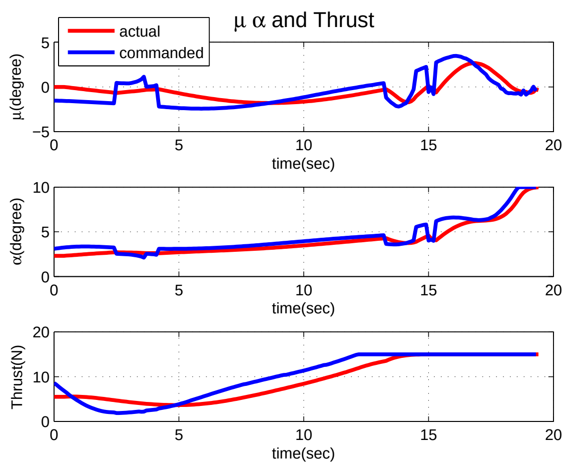


Figure 5.31: UAV Point mass model with EKF Multi Stationary Obstacle: χ, γ and Velocity

Figure 5.32: UAV Point mass model with EKF Multi Stationary Obstacle: μ, α and Thrust

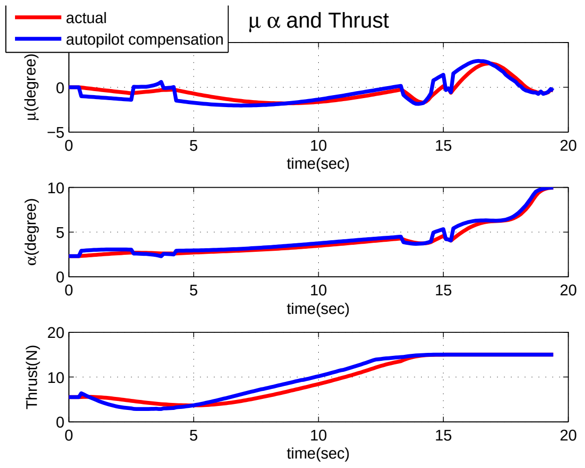


Figure 5.33: UAV Point mass model with EKF Multi Stationary Obstacle: Autopilot Compensation

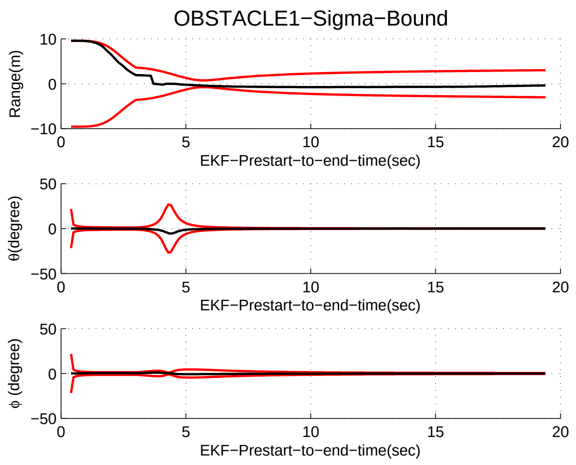


Figure 5.34: UAV Point mass model with EKF Multi Stationary Obstacle: Sigma error bound

The physical guidance commands α, μ and thrust actual and commanded profiles is shown in Fig. 5.32. The autopilot compensation is shown in Fig. 5.33. It can be observed that guidance with autopilot compensation tries to minimize the lag with the desired guidance command. The sigma error bound with respect to obstacle is shown in Fig. 5.34, the camera sensor states are well within bounds.

It is observed that the guidance command is efficiently able to perform the reactive collision avoidance in single stationary obstacle,single moving obstacle as well as multiple stationary obstacles scenario in the noisy sensor environments with EKF estimation of camera sensor states.

5.5 Point mass model of UAV with UKF Estimation

In this section Point mass model of UAV is considered and camera sensor model is assumed noisy and the stereovision camera sensor states estimation is carried out using Unscented Kalman Filter(UKF). The UKF excellence over EKF for nonlinear systems in noisy conditions are exploited and the results are presented for single moving obstacle with constant velocity as well multiple stationary obstacles environments.

5.5.1 Single obstacle

The simulation is carried out with single obstacle moving with constant velocity. Obstacle velocity $[2, 0, 0]$, the Initial position of UAV is $[0, 0, 50]$, Initial Velocity $[19.95, 0.97, 0.99]$, Target Position $[389.90, -3.0, 103.9]$, Obstacle initial position $[159.9, -1.5, 62.9]$, Obstacle final position $[199.0, -1.5, 62.98]$ and UAV Final location $[386.7, -2.4, 103.3]$. Point mass model of UAV is considered and unscented kalman filter is used for estimation. Fig. 5.35 shows the trajectory of the UAV. The trajectory plot is obtained through actual guidance as well as guidance through autopilot compensation. The guidance commands for the UAV with point mass model is shown in Fig. 5.45 where actual guidance command is tracking the desired guidance command.

The physical guidance command is shown in Fig. 5.46 with actual and commanded values of α, μ and thrust. The autopilot compensation for physical guidance command is shown in Fig. 5.38.

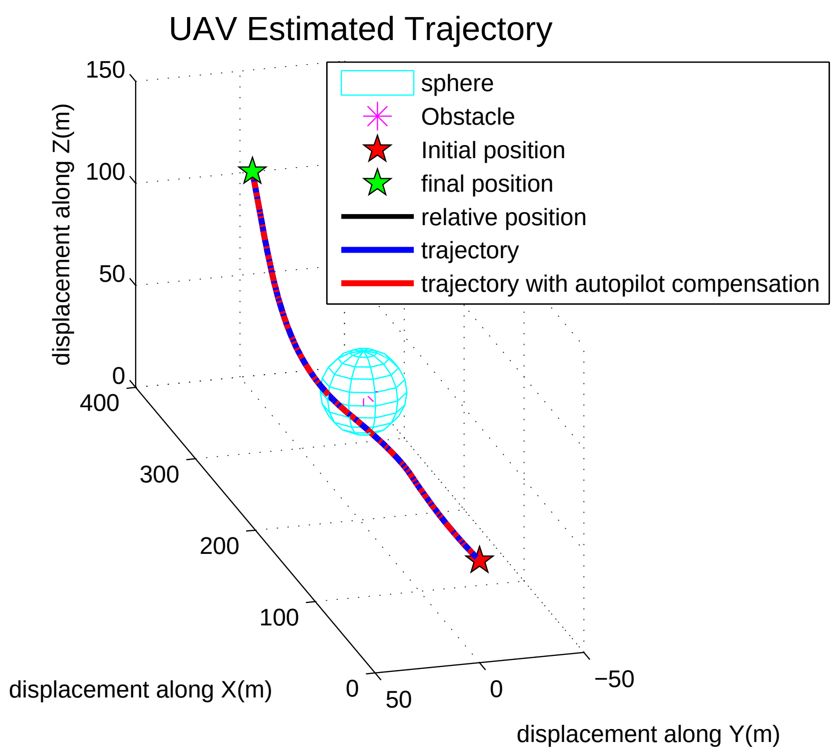
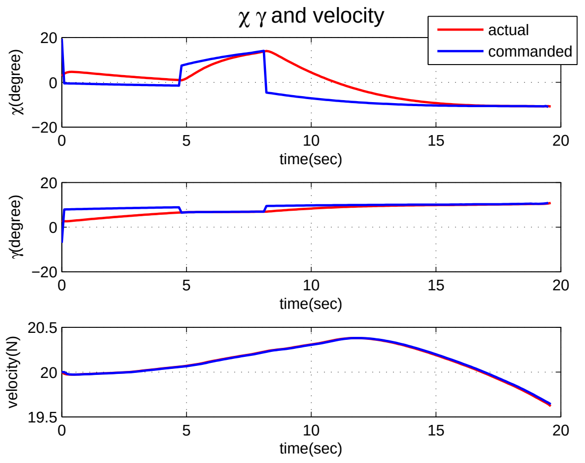


Figure 5.35: UAV Point mass model with UKF Single Moving Obstacle: UAV Trajectory



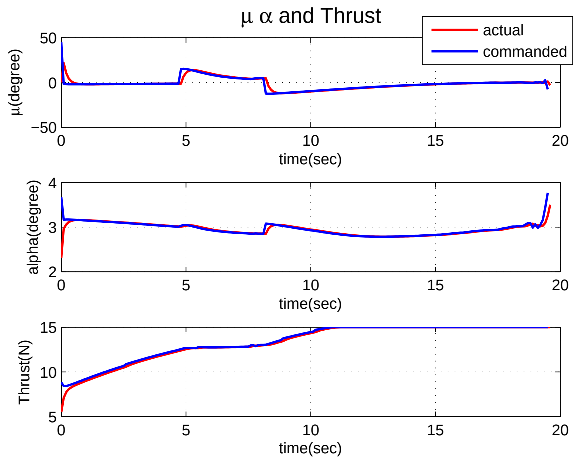


Figure 5.36: UAV Point mass model with UKF Single Moving Obstacle: χ, γ Velocity

Figure 5.37: UAV Point mass model with UKF Single Moving Obstacle: μ, α and Thrust

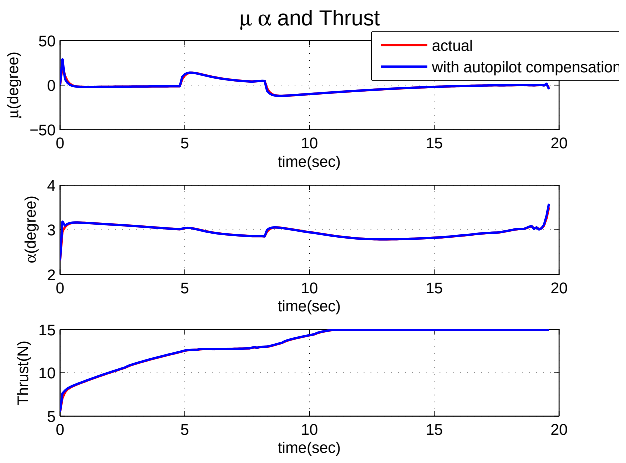


Figure 5.38: UAV Point mass model with UKF Single Moving Obstacle: Autopilot Compensation

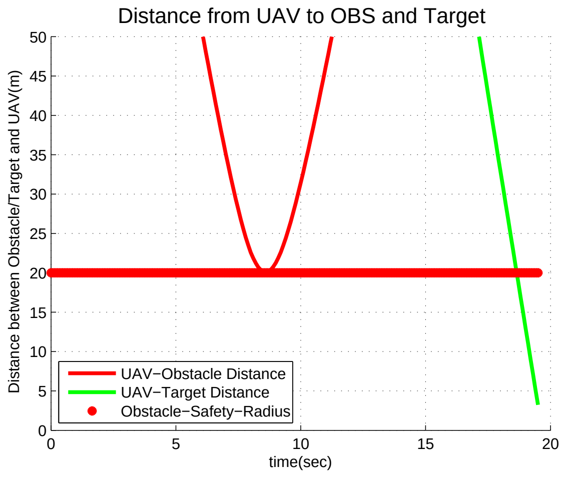


Figure 5.39: UAV Point mass model with UKF Single Moving Obstacle: UAV to Obstacle and Target Distance

The UAV to target distance and the UAV to obstacle distance profiles are shown in Fig. 5.39.

The obstacle is assumed to be moving with constant velocity and based on successive data obtained from camera sensors in their respective coordinate frames, using sensor state dynamics and using the unscented kalman filter estimation, the velocity of the obstacle is estimated and as shown in the Fig.5.40. The velocity estimate becomes erroneous as the camera sensor comes to closer to obstacle. This can be correlated with the time instants shown in the Fig.5.41 where the forward direction(x-direction) position components of UAV and obstacle are plotted.

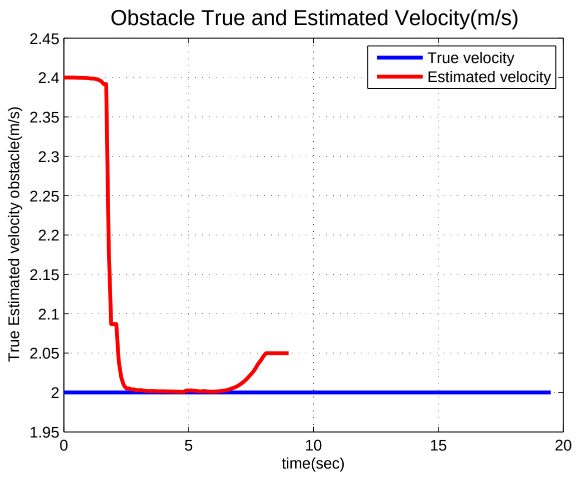


Figure 5.40: UAV Point mass model with UKF Single Moving Obstacle: Obstacle True and Estimated Velocity

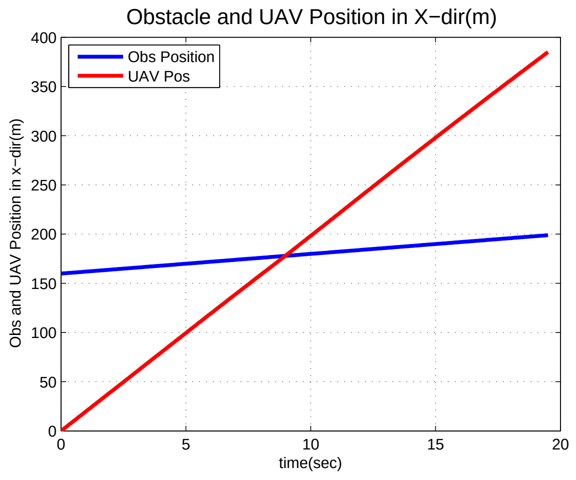


Figure 5.41: UAV Point mass model with UKF Single Moving Obstacle: Obstacle and UAV Position in x-direction

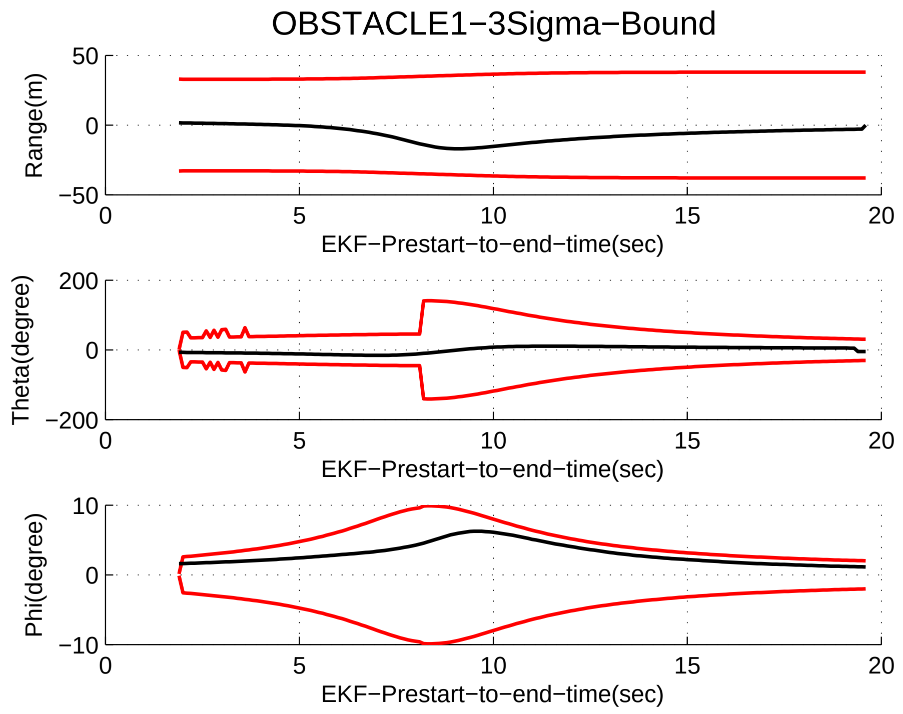


Figure 5.42: UAV Point mass model with UKF Single Moving Obstacle: Sigma error bound

The sigma error bound for stereovision camera sensor states is shown in Fig. 5.42. The camera sensor states are well with in bounds.

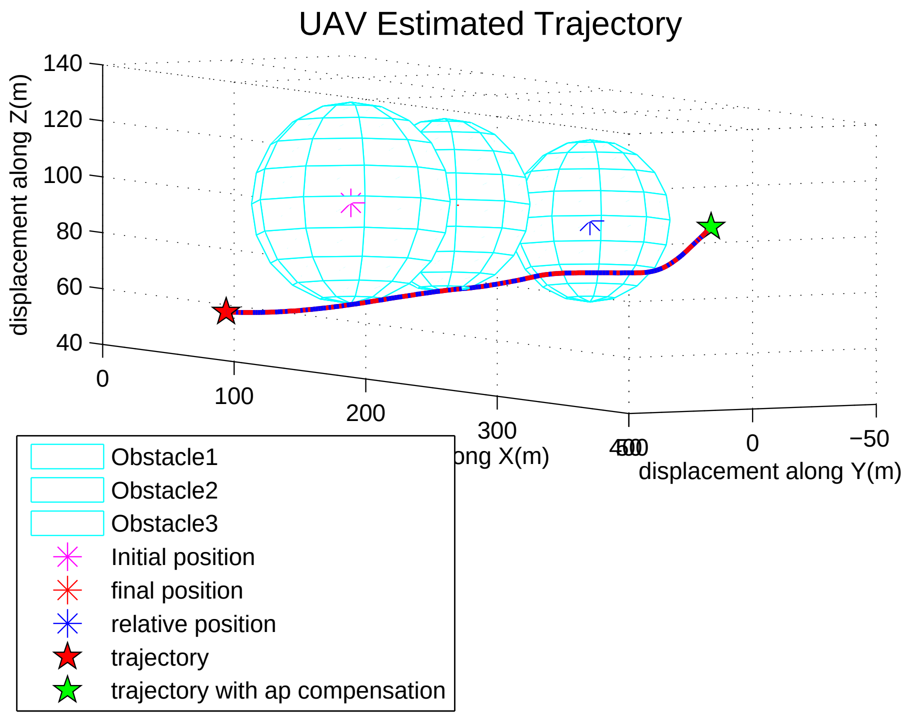


Figure 5.43: UAV Point mass model with UKF Multi Stationary Obstacle: UAV Trajectory

5.5.2 Multiple obstacle

In order to test the performance of reactive collision avoidance algorithm in multiple stationary obstacle scenario, the simulation is carried out with three stationary obstacles with Initial positions [89.9,-2.5,94.4],[179.9,7.48,99.49] and [299.9,12.48,101.49] the initial position of UAV is [0,0,50], Initial Velocity of UAV [19.95,0.97,0.99], Target Position [359.9,-4.5,102.45], UAV final position [356.23,-4.8,100.68]. point mass model of UAV is considered with UKF Estimation. Fig. 5.43 shows the trajectory of the UAV obtained through actual guidance command as well as guidance with autopilot compensation. In order to provide a clear view of UAV's closest approach with the three obstacles, the UAV to target distance profile and the UAV to three different obstacles distance profile is shown in Fig. 5.44. The guidance commands for the UAV with point mass model is shown in fig. 5.45 with actual and commanded guidance profiles for χ,γ and velocity.

The physical guidance commands α, μ and thrust actual and commanded profiles is shown in Fig. 5.46. The autopilot compensation for physical guidance command is shown in Fig. 5.47.

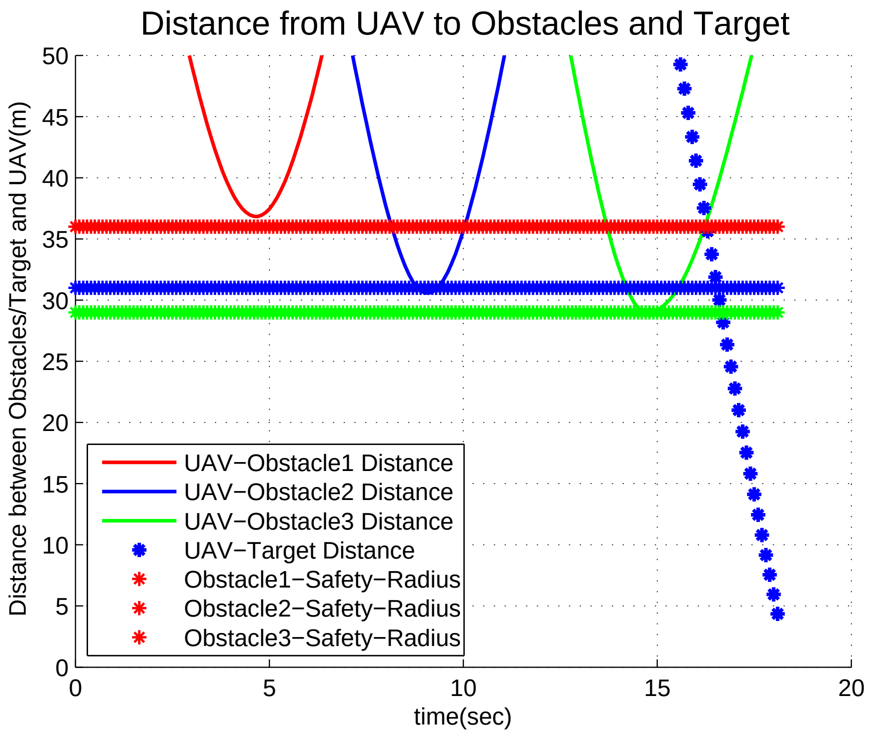


Figure 5.44: UAV Point mass model with UKF Multi Stationary Obstacle: UAV to Obstacles and Target Distance

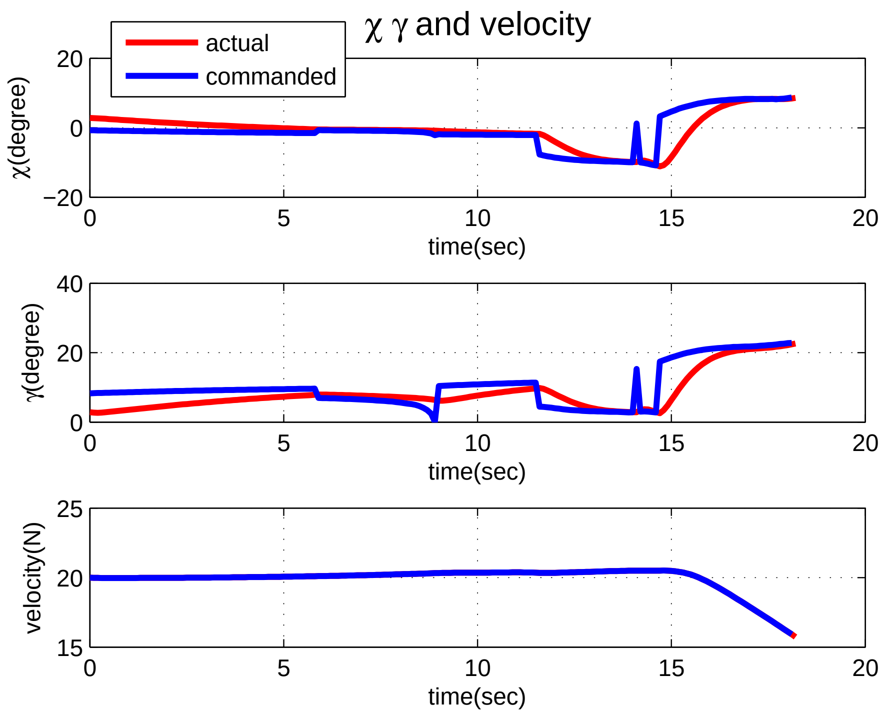


Figure 5.45: UAV Point mass model with UKF Multi Stationary Obstacle: χ, γ and Velocity

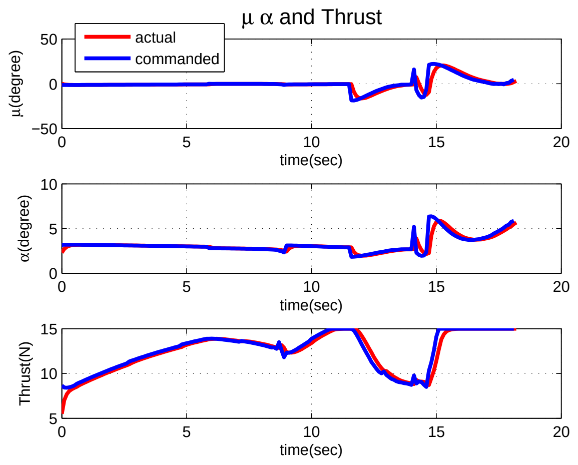


Figure 5.46: UAV Point mass model with UKF Multi Stationary Obstacle: μ, α and Thrust

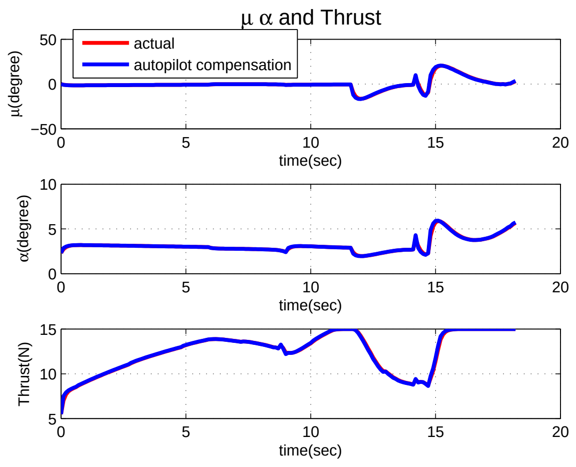


Figure 5.47: UAV Point mass model with UKF Multi Stationary Obstacle: Autopilot Compensation

It is observed that the guidance command is efficiently able to perform the reactive collision avoidance for single moving obstacle as well as multiple stationary obstacles scenario in the noisy sensor environments with UKF estimation of camera sensor states.

5.5.3 Randomized Validation

In order to test the performance of the reactive collision avoidance algorithm in different randomized initial conditions, simulations are carried out for single moving obstacle. Point mass model of UAV is used for system dynamics and UKF is used for camera sensor state estimation. The parameters randomly chosen are UAV initial velocity and initial position, obstacle initial position and destination. 1000 randomized simulation runs are carried out. The obstacle position is estimated using UKF and the effectiveness of nonlinear DGG guidance law is tested. Fig. (5.48) shows the success criteria of proposed algorithm in different success bands S1, S2, S3 and S4. The percentage of success for different bands are tabulated in Table 5.1.

Table 5.1: Success Percentage with Obstacle Estimation

Success Band	Tolerable safety sphere violation (as % of the safety sphere radius)	Success percentage
S1-Band	10 %	86 %
S2-Band	20 %	99.2 %
S3-Band	30 %	100 %
S4-Band	40 %	100 %

Validation results show that reactive collision avoidance algorithm is quite effective and reliable.

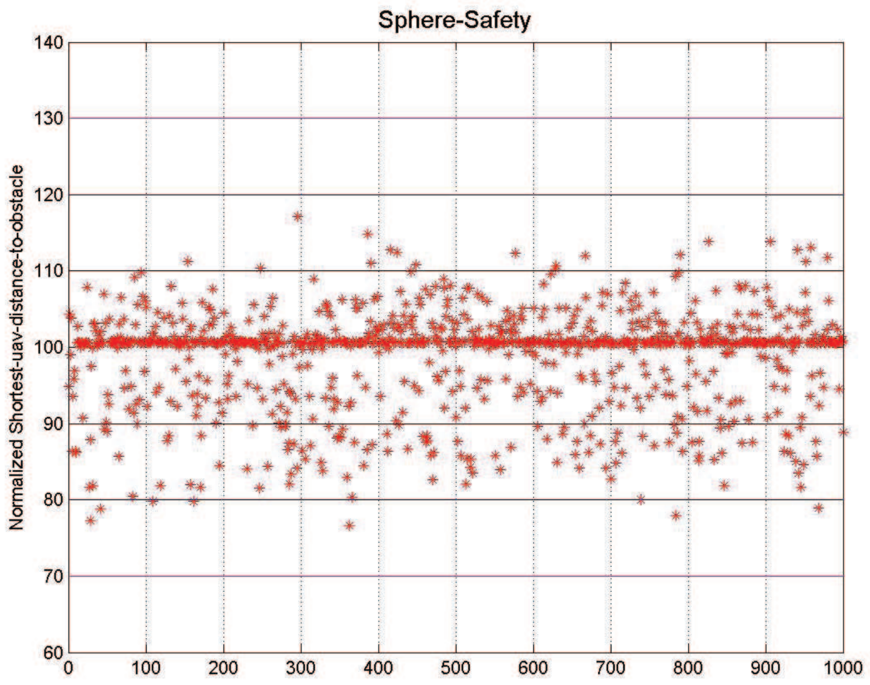


Figure 5.48: UAVs Closest Approach to Obstacle for 1000 runs

Chapter 6

Acknowledgements

This work is supported by AOARD/AFRL, USA, under the contract FA-2386-11-1-4096, which operated at the Society for Innovation and Development (SID), Indian Institute of Science, Bangalore, India, with the project code SID/PC 99203.

Chapter 7

Conclusion

UAVs are playing a vital role in numerous applications. Even in the areas which are inaccessible to human beings, UAVs can outperform with onboard vision sensors and in-built processors. The present work focuses on the reactive obstacle avoidance problem for unaccountable stationary obstacles like urban edifices, poles as well as moving obstacles like another UAV, flying birds etc.

The guidance strategy applied in the kinematic model as well as point mass model based formulation uses the collision cone approach for obstacle detection and executes the avoidance maneuver by generating the angular guidance commands in the horizontal and the vertical planes. UAV pursues the guidance commands by quickly aligning its velocity vector along the aiming point while enforcing the turn coordination in the case of point mass model. The obstacle position as well as velocity can be considered to be partially known with some noise and hence can be estimated by using the EKF and UKF techniques. In all the simulations, all the constraints posed by the vehicle capability are very well met within the available time-to-go.

The proposed Formulation can be enhanced by augmenting it with features like introducing moving obstacles with specific maneuvers along with the stationary obstacles. Instead of spherical safety zones around the obstacles, more optimized shape of the safety zone like cylinder can be considered to accommodate obstacles like electric poles and wires. The obstacle information can also be processed through real passive sensors like cameras which may involve wide exploration in the computer vision techniques.

Reactive maneuvers for obstacle avoidance demands guidance and control to execute in synergy. Test cases for the point mass model with a coordinated flight and kinematic model is demonstrated with all scenarios, where the controller dynamics were approximated as the first order autopilots. Using the stereovision camera, an effective reactive collision avoidance algorithm is presented in this report. The following assumptions are made, (i) the shape and size of the obstacle is known and (ii) the data collected from the vision sensors are assumed to be noisy. So the camera captured position information may become noisy, to overcome this, two standard filtering techniques such as Extended Kalman Filter(EKF) and Unscented Kalman Filter(UKF) are proposed to extract the useful information about the obstacle position from the noisy camera data. Obstacle velocity information is also derived from the second order sensor dynamics and estimated. However in later portion of the report, virtual simulation is carried out for spherical obstacle by extracting the potential feature points from stereo images using harris detector and then, stereo matching is done along the epipolor line. Finally, the 3D information of obstacle obtained by applying triangulation method to the matched feature points. Next, following the collision cone approach, an 'aiming point' is computed and nonlinear differential geometric guidance is applied to steer away the vehicle quickly from the impending danger. Simulation studies leads to the conclusion that this strategy is quite effective in avoiding popup obstacles within a very short time and hence can be very useful for reactive collision avoidance. This algorithm generalized to include (i) moving obstacles with simple maneuvers, (ii) more than one obstacle at the same time and (iii) appropriate guidance of the vehicle after reaching the aiming point.

Bibliography

[1] Choi, H. and Kim, Y., "Reactive Collision Avoidance of Unmanned Aerial Vehicles Using a Single Vision Sensor", *Journal of Guidance, Control and Dynamics*, Vol.36,No.4,July-August 2013.

[2] Williams, S. P., Antoniewicz, R. F., Duke, E. L. and Menon, P. K. A., "Study of a Pursuit-Evasion Guidance Law for High Performance Aircraft", *Proceedings of American Control Conference, Pittsburg, PA, USA*, 21-23 June, 1989.

[3] D. Simon, Optical State Estimation, *John Wiley and Sons*, New Jersey,2006.

[4] S. Haykin, Kalman Filtering and Neural Networks, *John Wiley and Sons*, 2001.

[5] Grillo, C. and Vitrano, P. F., "State Estimation of a Nonlinear Unmanned Ariel Vehicle Model using an Extended Kalman Filter", *15th AIAA International Space Planes and Hypersonic Systems and Technologies Conference, Dayton, Ohio*, 28 April-1 May, 2008.

[6] Mujumdar, A. and Padhi, R., "Evolving Philosophies on Autonomous Obstacle/Collision Avoidance of Unmanned Aerial Vehicles", *AIAA Journal of Aerospace Computing, Information and Control*, 2011a, Vol.8, pp.17–41.

[7] Mujumdar, A. and Padhi, R., "Reactive Collision Avoidance Using Nonlinear Geometric and Differential Geometric Guidance", *AIAA Journal of Guidance, Control and Dynamics*, 2011b, Vol.34, No.1, pp.303–311.

[8] Lu-Ping, Tsao, Ching-Lain, Chou, Chuen-Ming, Chen, and Chi-Teh (1998), "Aiming Point Guidance Law for Air-to-Air Missiles", *International Journal of Systems Science*, Vol. 29, No. 2, 1998, pp.95-102.

[9] Padhi, R. and Gupta, A., "Dynamic Estimation of Obstacle Position with Vision Sensing for Reactive Collision Avoidance of UAVs", *Journal of Aerospace Sciences and Technologies*, 2012, Vol.64, No.3, pp.187–200.

[10] Tripathi, A.K.,Raja R.G., and Padhi, R., "Reactive Collision Aviodance of UAVs with Vision Sensing using Pin-hole Cameras", *IFAC Symposium on Automatic Control in Aerospace*, Wurzburg , Germany, September 2–6,2013.

[11] Hrabar, S., Sukhatme, G. S., Corke, P., Usher, K., and Roberts, J., "Combined optic-flow and stereo-based navigation of urban canyons for a UAV", *In Proceedings of IEEE International Conference on Intelligent Robots and Systems, Alberta*, 2004, pages 3609 3615.

[12] LaValle, S. M., and Kuffner, J. J., "Rapidly-exploring random trees: Progress and prospects", *Algorithmic and Computational Robotics: New Directions*, 2001.

[13] Scherer, S., Singh, S., Chamberlain, L., and Elgersma, M., "Flying Fast and Low Among Obstacles: Methodology and Experiments", *The International Journal of Robotics Research*, 2008, Vol. 27, No. 5, pages 549 574.

[14] Shim, D. H., Chung, H., and Sastry, S., "Autonomous Exploration in Unknown Urban Environments for Unmanned Aerial Vehicles", *IEEE Robotics and Automation Magazine*, 2006, Vol. 13, No. 3, pages 27 - 33.

[15] Hwangbo, M., Kuffner, J., and Kanade, T., "Efficient Two-phase 3D Motion Planning for Small Fixed-wing UAVs", *IEEE International Conference on Robotics and Automation*, 10 - 14 April 2007, Rome, Italy.

[16] Han, S. C., and Bang, H., "Proportional Navigation-Based Collision Avoidance for UAVs", *International Journal of Control, Automation and Systems*, 2009, Vol.7, No.4, pages 553 - 565.

[17] Watanabe, Y., Calise, A. J., and Johnson, E. N., "Minimum Effort Guidance for Vision-Based Collision Avoidance", *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, 21 - 24 August 2006, Keystone, Colorado.

[18] Carbone, C., Ciniglio, U., Corraro, F., and Luongo, S., “A Novel 3D Geometric Algorithm for Aircraft Autonomous Collision Avoidance”, *Proceedings of the* 45th *IEEE Conference on Decision and Control*, 13 - 15 December 2006, San Diego, CA, USA.

[19] Park, J. W., Oh, H. D., and Tahk, M. J., “UAV Collision Avoidance Based on Geometric Approach”, *SICE Annual Conference*, 2008, pages 2122 - 2126.

[20] Hull, D. G., “Fundamentals of Airplane Flight Mechanics”, *Springer*, 2007.

[21] R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision”, *Cambridge University Press*, 2000.

[22] Sergiu Nedevschi, Radu Danescu and C. Pocol, “High accuracy stereo vision system for far distance obstacle detection, *in 2004 IEEE intelligent vehicle symposium*, university of parama,2004.

[23] D. G. Lowe, “Object recognition from local scale-invariant features”, *Int. Conf. Computer Vision*, Corfu, Greece, September 1999, pp. 1150-1157.

[24] D. G. Lowe, “Distinctive image features from scale-invariant keypoints”, *Int. J. Comput. Vis.* vol. 60 no.2, pp. 91-110, 2004.

[25] H. Bay, A. Ess, T. Tuytelaars, L. van Gool, “Speeded-up robust features (SURF)”, *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346359, 2008.

[26] E. Rosten, T. Drummond, “Machine learning for high-speed corner detection”, *European Confer- ence on Computer Vision*, pp. 430443, 2006.

[27] Zhiwei Zhu, Taragay Oskiper and R. Kumar, An improved stereo-based visual odometry system, *Sarnoff Corpo- ration 201* Washington Road, Princeton, NJ 08540, USA.

[28] C. Harris and M. Stephens, A combined corner and edge detector, *4th Alvey Vision Conf*, Manchester, 1988

[29] Mujumdar A., and Padhi, R.,“Nonlinear Geometric and Diferential Geometric Guidance of UAVs for Reactive Collision Avoidance”,*AOARD Project, 2010, Project Ref. No. IISc/SID/AE/LINCGODS/AOARD/2010/01*,Department of Aerospace Engineering, Indian Institute of Science, Bangalore.

[30] Xin, M., Balakrishnan, S. N., Stansbery, D. T., and Ohlmeyer, E. J., “Nonlinear Missile Autopilot Design with $\theta - D$ Technique”, *Journal of Guidance, Control, and Dynamics*, Vol.27, No.3, May-June 2004.

[31] Chawla, C., and Padhi, R., “Reactive Obstacle Avoidance of UAVs with Dynamic Inversion Based Partial Integrated Guidance and Control”, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2 - 5 August 2010, Toronto, Canada.

[32] Stevens, B., and Lewis, F., “Aircraft Control and Simulation 2^{nd} Edition”, *J.Wiley & Sons*, 2003.

[33] Shenoydor, N. A., “Missile Guidance and Pursuit: Kinematics, Dynamics and Control”,*Horwood Publishing Limited*, 1998.

[34] Tsao, P. L., Chou, C. L., Chen, C. M., and Chen, C. T., “Aiming Point Guidance Law for Air-to-Air Missiles”, *International Journal of Systems Science*, 1998, Vol. 29, No. 2, pages 95 - 102.

[35] Enns, D., Bugajski, D., Hendrick, R., and Stein, G., “Dynamic inversion: an evolving methodology for flight control design”, *International Journal of Control*, 1994, Vol. 59, No. 1, pages 71 - 91.

[36] Singh, S. P., and Padhi, R., “Automatic Path Planning and Control Design for Autonomous Landing of UAVs using Dynamic Inversion”, *American Control Conference*, 10 - 12 June 2009, St. Louis, USA.

[37] Singh, S. P., “Autonomous Landing of Unmanned Air Vehicles”, Department of Aerospace Engineering, Indian Institute of Science, Bangalore, 2009.

[38] Beard, R., Kingston, D., Quigley, M., Snyder, D., Christiansen, R., Johnson, W., McLain, T., and Goodrich, M. A., “Autonomous Vehicle Technologies for Small Fixed-Wing UAVs”, Journal of Aerospace Computing, Information, and Communication, Vol. 2, January 2005.

[39] Surendra nath, V., Govindaraju, S. P., Bhat, M. S., and Rao, C. S. N., “Configuration Development of All Electric Mini Airplane”, *ADE/DRDO Project, 2004, Project Ref. No: ADEO/MAE/VSU/001*, Department of Aerospace Engineering, Indian Institute of Science, Bangalore.

[40] Atkinson, K. E.,"An Introduction to Numerical Analysis",*John Wiley & Sons*, 2001.

[41] Chakravarthy, A., and Ghose, D., "Obstacle Avoidance in a Dynamic Environment: A Collision Cone Approach", *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans*, 1998, Vol. 28, No. 1, pages 562 - 574.

[42] Pollefeys M., "Self Calibration and 3D Reconstruction from Uncalibrated Image Sequences", *PhD Thesis Report,Katholieke Universiteit Leuven,3001 Heverlee, Belgium*, 1999.

[43] Haoxiang L., Muhammad T. K.,Kok-Kiong T., C. W. deSilva,"Developments in Visual Servoing for Mobile Manipulation", *Unmanned Systems*, Vol. 1, No. 1, 1-20,2013.

[44] Sabestian G.,"Camera Pose Estimation from a Stereo Setup", *Thesis Report,University of Ottawa,Ontario,Canada*, Feb.2005.

[45] Roberto Sabatini R., Richardson M.,Bartel C.,Shaid T. and Ramasamy S., "A Low-cost Vision Based Navigation System for Small Size Unmanned Aerial Vehicle Applications", *J. of Aeronaut Aerospace Eng*, May, 2013.

[46] Wikus B.,"Stereo vision for simultaneous localization and mapping", *Thesis Report,Stellenbosch University*, December, 2012.

[47] Ravikumar L.,Philip N.K., Bhat M.S. and Padhi R.,"Object detection and obstacle avoidance for rover using stereo camera", *IEEE-MSC Conference, Hyderabad, India*, August, 2013.

[48] Raja R.G., Chawla C. and Padhi R.,"Dynamic Inversion-Based Nonlinear Aiming Point Guidance of Unmanned Aerial Vehicles for Reactive Obstacle Avoidance", *Unmanned Systems*, Vol. 1, No. 2 (2013) 259275.

Appendix A

Mathematical Model

In this report, two standard system dynamics for the vehicle movement have been considered. First, a simplistic ”kinematic model” has been considered with autopilot lags in all channels of the state equation for quick validation of the basic philosophy. Subsequently, to be more realistic, the proposed guidance strategy has been reworked using a ”point mass model” of the vehicle and further algebra has been carried out by introducing an additional inner loop to generate the physically meaningful guidance parameters, namely the desired bank angle, angle of attack and the thrust required for the vehicle, as the necessary guidance parameters. Note that to be even more realistic, autopilot lags for these quantities have been assumed as well while validating our proposed guidance strategy, which will be discussed later. In this section, the relevant details of the mathematical models of both kinematic as well as point mass models used in this research are given in fair detail.

A.1 Kinematic Model

The relevant set of equations in this model are given by

$$\dot{x} = V_T \cos\chi \cos\gamma \tag{A.1}$$

$$\dot{y} = V_T \sin\chi \cos\gamma \tag{A.2}$$

$$\dot{h} = V_T \sin\gamma \tag{A.3}$$

$$\dot{V}_T = k_v(V_T^c - V_T) \tag{A.4}$$

$$\dot{\gamma} = k_\gamma(\gamma^c - \gamma) \tag{A.5}$$

$$\dot{\chi} = k_\chi(\chi^c - \chi) \tag{A.6}$$

where, V_T is the velocity of the UAV whose directions are given by two angles, namely the flight path angle γ and course angle χ. The first three equations of Eq.(A.1-A.3) represent the movement of the centre of mass of the vehicle in an inertial frame depending on the velocity vector magnitude and its orientation. The next three equations essentially represent first order lags, which in turn govern the evolution of V_T, γ and χ, based on their commanded values V_T^c, γ^c and χ^c respectively. The key idea here is to generate γ^c and χ^c from an appropriate guidance formulation, as well as giving an appropriate V_T^c command, so that the obstacles can be avoided.

A.2 Point Mass Model

To achieve obstacle avoidance in a more realistic environment, the vehicle is considered as a point mass object. V_T is the inertial velocity of the UAV whose directions are given by two angles, course angle(χ) and flight path angle(γ). Hence, the angular corrections on the point mass are realized by the basic aerodynamic forces acting on the vehicle. The 6^{th} order model representing the coordinated, symmetric flight over a flat earth (with standard meaning of the variables [20]) is considered. In the coordinated flight, the side force acting on the vehicle is null and hence the side slip angle $\beta = 0$ [32].

$$\begin{aligned}
\dot{x} &= V_T \cos\chi \cos\gamma && \text{(A.7)} \\
\dot{y} &= V_T \sin\chi \cos\gamma && \text{(A.8)} \\
\dot{h} &= V_T \sin\gamma && \text{(A.9)} \\
\dot{\chi} &= \frac{1}{mV_T \cos\gamma} (T \sin\alpha + L) \sin\mu && \text{(A.10)} \\
\dot{\gamma} &= \frac{1}{mV_T} [(T \sin\alpha + L) \cos\mu - mg \cos\gamma] && \text{(A.11)} \\
\dot{V}_T &= \frac{1}{m} (T \cos\alpha - D - mg \sin\gamma) && \text{(A.12)}
\end{aligned}$$

Therefore, the only forces acting on the vehicle are thrust, lift, drag and weight. To model the aerodynamic forces, aerodynamic data of a prototype UAV, named as all electric airplane-2 $(AE-2)$ [39] is used.

Figure A.1: AE-2 (Picture of All Electric airplane-2)

This UAV has a reference area of $S = 0.6m^2$ and it has mass $6kg$. Maximum value of thrust (T_{max}) which can be produced by its electric motor and propeller assembly is $15N$. It is assumed that thrust produced has linear relation with the throttle input. Forces L and D acting on the vehicle are given by

$$L = \bar{q}SC_L \quad D = \bar{q}SC_D \tag{A.13}$$

$\bar{q}$ is the dynamic pressure,C_L and C_D are the lift and drag coefficients in wind axes system which can be expressed in terms of axial and normal force coefficients C_X and C_Z of the body axes system.

$$\begin{bmatrix} -D \\ -L \end{bmatrix} = mL_{W/B} \begin{bmatrix} X_a \\ Z_a \end{bmatrix} \tag{A.14}$$

$X_a,\ Z_a$ are the body axes accelerations, $L_{W/B}$ is the transformation matrix from body to wind axes system and is given by

$$L_{W/B} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}$$

Note that Eq. (A.14) assumes that the side-slip angle is zero. This is very close to reality with the assumption of co-ordinated turn, which is ensured in the formulation while generating the necessary bank angle as the set of equations in Eq.(A.10-A.12) are valid only under this assumption. By expanding Eq. (A.14), drag and lift forces can be expressed in terms of body axes axial and normal force coefficients as follows

$$D = -\bar{q}S\left[(C_{X_0} + C_{X_{a1}}\alpha + C_{X_{a2}}\alpha^2)\cos\alpha + (C_{Z_0} + C_{Z_{a1}}\alpha)\sin\alpha\right]$$
$$L = -\bar{q}S\left[(C_{X_0} + C_{X_{a1}}\alpha + C_{X_{a2}}\alpha^2)(-\sin\alpha) + (C_{Z_0} + C_{Z_{a1}}\alpha)\cos\alpha\right]$$

Where, C_{X_0} and C_{Z_0} are the aerodynamic coefficients and $C_{Z_{a1}}$,$C_{X_{a1}}$,$C_{X_{a2}}$ are the dynamic derivatives of the aerodynamic coefficients. The initial values of the aerodynamic coefficients and their derivatives corresponding to the UAV model [39] are given in Table A.1.

Table A.1: Initial Value of Coefficients

C_{X_0}	C_{Z_0}	$C_{Z_{a1}}$	$C_{X_{a1}}$	$C_{X_{a2}}$
0.0386	0.1653	0.087138	-0.0040376	-0.0010525

Note that the ultimate aim here is to find the desired values of angle of attack α , bank angle σ and the desired thrust T that will ensure collision avoidance.

First order autopilots are designed for all the control variables which can be stated as

$$\dot{\mu}_d = -k_\mu(\mu_d - \mu^*) \tag{A.15}$$
$$\dot{\alpha}_d = -k_\alpha(\alpha_d - \alpha^*) \tag{A.16}$$
$$\dot{T}_d = -k_T(T_d - T^*) \tag{A.17}$$

μ^*, α^*, T^* are the control variables with full bandwidth and the μ_d, α_d, T_d are the control variables obtained after the first order delay of the autopilot. Gains for autopilot are much higher than the gains for the guidance loop due to difference in the settling time of the guidance and autopilot dynamics. The position limits on the control variables μ_d, α_d, T_d are given by $\mu_d \leq \pm 45^0$ (maximum turn angle), $\alpha_d \leq \pm 10^0$ (stall angle) and $T_d \leq \pm 15^0$ (maximum thrust limit).